AI Needs-Based Configurator Implemented in Java

By

Kenneth James Gabel Lynch

Submitted to the Department of Electrical Engineering and Computer Science
in Partial Fulfillment of the Requirements for the Degrees of
Bachelor of Science in Electrical Engineering and Computer Science
and
Master of Engineering in Electrical Engineering and Computer Science
at the
Massachusetts Institute of Technology
May, 1998
[June 1998]

Author_____

Department of Electrical Engineering and Computer Science
May 22, 1998

Certified by_____

Glen L. Urban
Dean, MIT Sloan
Thesis Supervisor

Accepted by_____

Arthur C. Smith
Chairman, Department Committee on Graduate Theses

AI Needs-Based Configurator Implemented in Java

By

Kenneth James Gabel Lynch

Submitted to the Department of Electrical Engineering and Computer Science
on May 22, 1998, in partial fulfillment of the
requirements for the Degrees of
Bachelor of Science in Electrical Engineering and Computer Science
and
Master of Engineering in Electrical Engineering and Computer Science

Abstract

This thesis describes the Java implementation and future extensions of a sales recommendation generator. The engine consists of truck knowledge databases, expert engines, an Internet GUI and is extended with user specific "consultative selling" behavior. The later half of the thesis describes possible extensions for future research. These extensions utilize the existing Java code-base to produce higher recommendation quality, higher user acceptance, and new product generation.

Thesis Supervisor: Glen L. Urban

Title: Dean, MIT Sloan

Table of Contents

Table of Figures

1. Thesis Introduction

This thesis describes the Java implementation and future extensions of a sales recommendation generator. This needs-based expert advisor is part of the Trucktown Java Internet program. It is created for the MIT Sloan Trust Based Marketing Project and sponsored by General Motors & IMUP. It consists of truck knowledge databases, expert engines, an Internet GUI and is extended with user specific "consultative selling" behavior. The expert advisor involves the implementation of a personable human GUI with underlying intelligence in the form of an updating Bayesian probabilistic network, market segmentation filtering, and truck utility analysis. The later half of the thesis describes possible extensions for future research. These extensions utilize the existing Java code-base to produce higher recommendation quality, higher user acceptance, and new product generation.

2. Thesis Goals and Summary

2.1 Goals of this Research

1. Develop analytic structure and GUI for the expert advisor, exploring the commercial context of a recommendation generator.
2. Extend the recommendation engine to include "consultative selling" behavior.
3. Evaluate the methodology involved in the Java Internet implementation of this knowledge-based system.
4. Explore possible extensions and alternative courses of development for further research.

2.2 Summary

The greater Trucktown Internet Site environment is designed to research how to develop users' trust, customer relationship and business loyalty. As part of this "Trucktown experience", the user is led by a trustworthy guide to a human expert advisor. This advisor is the GUI for the needs-based configuration engine. The advisor builds the customer relationship while collecting information to be used in the sale of the recommended trucks.

The underlying logic of the expert system is abstracted into several expert objects working in collaboration with each other. These experts are abstracted into a De-Correlation Expert, Utility Analysis Expert, Bayesian Expert, and Market Segmentation Expert.

The system implements the concept of recommendation explanation, or "consultative selling" behavior. It is implemented by correlating sales features and problems with user specific actions and responses. The purpose is to relate the underlying logic to the user in

an understandable dialog and help the marketing expert catch otherwise non-obvious logic problems.

The Java Internet issues investigated are problems specific to an Internet client-side multimedia environment. These programming issues encompass two main topics:

1. Implementation of the expert GUI
2. Implementation of the underlying logic and how it is abstracted and manipulated.

The research conclusion is that the Java environment was found to be excellent for managing knowledge based manipulation but still poor for client side GUI manipulation.

There are four main ways the expert system research can be extended:

1. Improve recommendation quality
   a. Real expert evaluation and refinement of questions and truck data
   b. Benchmark quality with real and simulated users
   c. Utilize more sophisticated algorithms such as an Inconsistency Expert
2. Use the expert system for new product development
3. Further integrate the recommendation engine with the sales process
   a. Improve the GUI
   b. More dynamic customization of sales process, extending the Consultative Sales Expert
   c. Integrate with an underlying specification-based configuration engine
4. Generalize the engine to sell other products

3. General Implementation of the Expert System

This expert system is part of a more general study created for the MIT Sloan Trust Based Marketing Project. One of the major goals of this study is to create a trusting environment. As part of this environment, the customer uses a human advisor GUI interface to filter information and product recommendations for the user. The underlying logic is a collection of expert algorithms taking the information to produce user specific recommendations and explanations.

3.1.1 Trucktown Main Environment

The main Trucktown environment was created to test consumer trust in a general marketing study. The user navigates a main map with the optional help of a cartoon guide. Hot spots (or links) on the main map trigger the creation and showing of a module. The Expert/Advisor Object is one such module. The main environment provides general services such as image retrieval. The main environment also keeps user information and history.

3.1.2 Trucktown Functionality

Trucktown is a navigation concept meant to allow more freedom and more closely resemble the traditional buying experience. An owl guide directs the user to places of interest. The user can click to visit any of the three experts and visit the showroom. The showroom module knows about the conversation with the last expert, displays the highest ranked trucks and provides information regarding those truck recommendations.

The guide (implemented as an owl) is aware of which modules the user has visited and will direct the dialog and user choices according to previous user navigation. For example, if the user attempts to go to the showroom without having completed a visit with an expert, the owl will suggest another visit to an expert instead of going to the showroom.

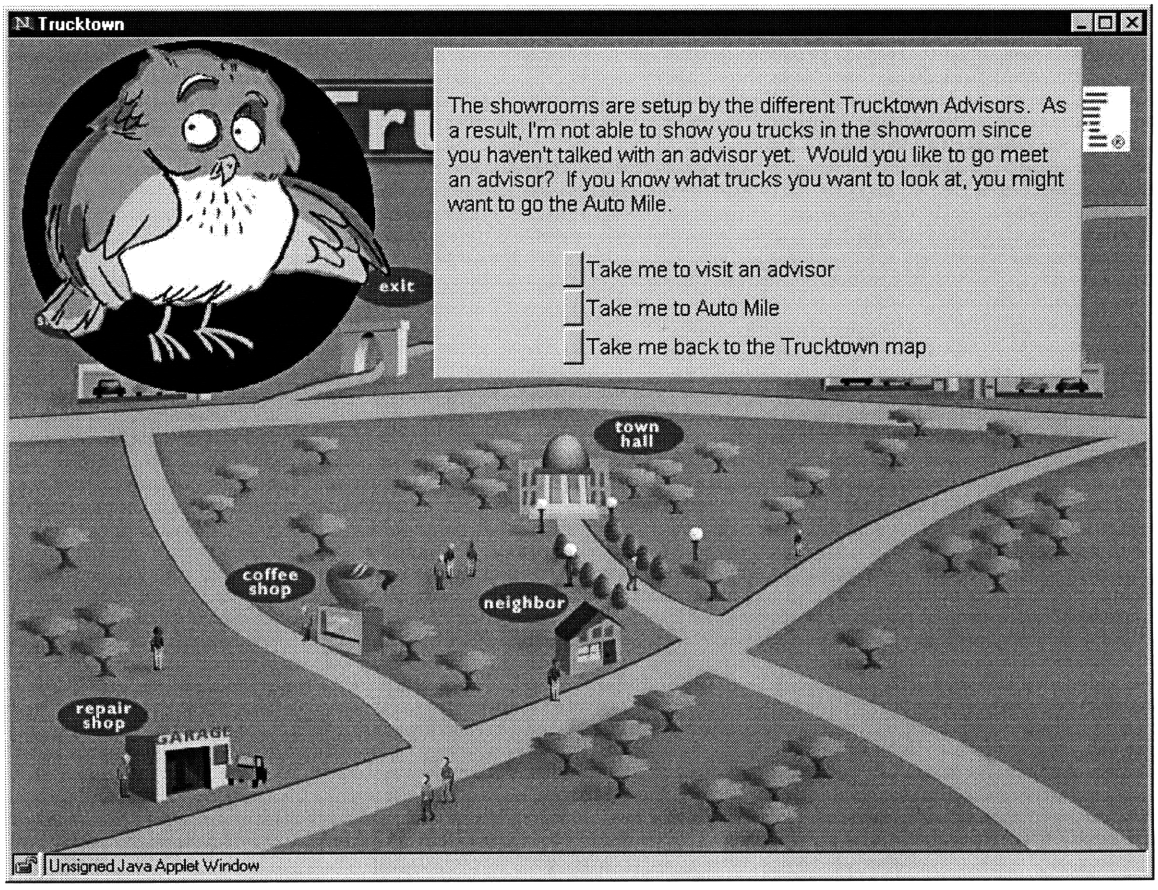


Figure 1. Main Trucktown Map with Owl Guide

3.1.3 Trucktown High Level Architecture

The general Trucktown architecture is object oriented. All modules are distinct objects with Inter-module communication occurring through the Session Module object. Modules are not allowed to communicate directly with each. Instead, they communicate exclusively though a Trucktown Frame Interface. This abstraction barrier allows for an extremely smooth integration process with multiple independent programmers working on separate modules and tasks.

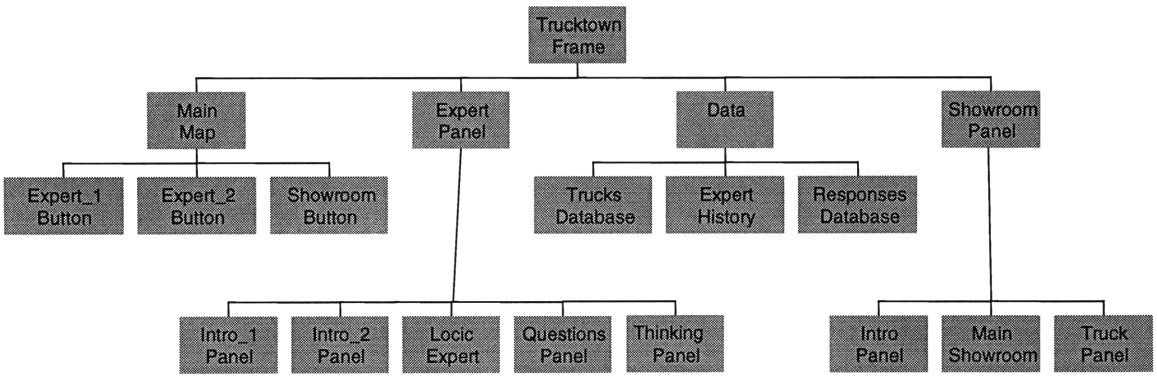


Figure 2. Trucktown Object Diagram

There are three instanciations of the Expert Panel object. Upon construction, these Expert Panels are passed a variable to invoke separate GUIs. The corresponding "Top Layer Response Database" for each expert is stored in "data" (an instanciation of SessionModal). The responsesDB (the user response database from the expert package) is also stored in "data". The experts can communicate their recommendations to the showroom through "data" as well. In practice, the responsesDB is passed to the showroom through "data" and a private Logic Expert is instanciated to produce recommendations. This private instanciation ensures that there will be a current recommendation set. This methodology is more robust and easier to maintain due to their being less intermediate states and no requirements for intermediate calculations. All intermediate recommendations are **re-calculated every time.**

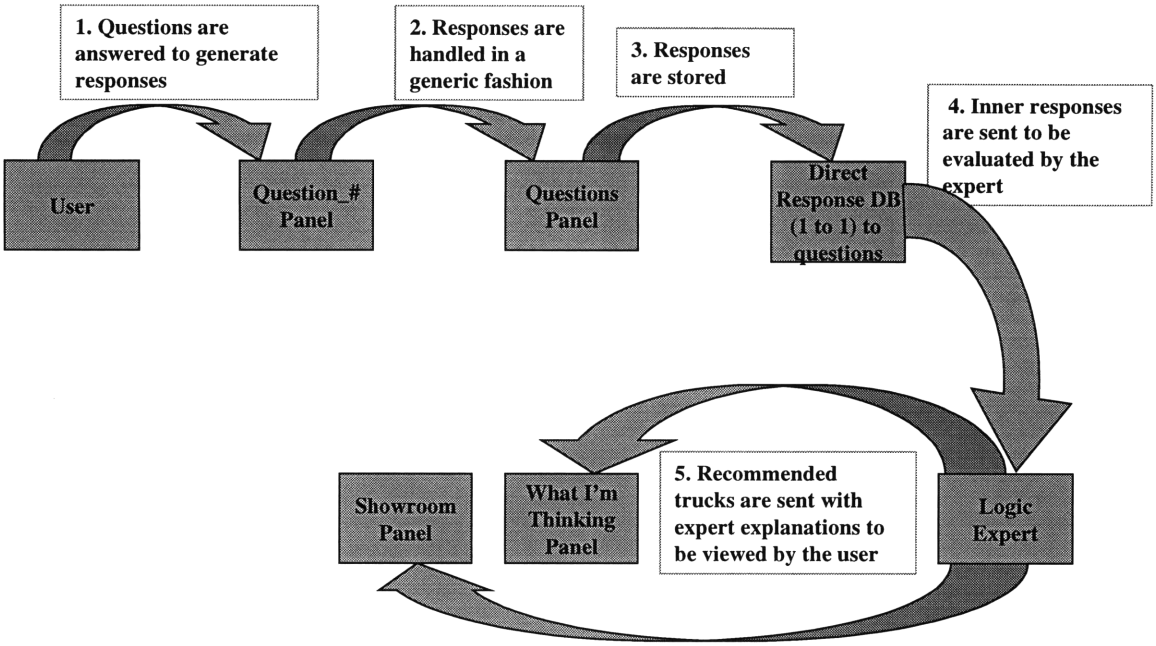


Figure 3. General Data Flow Diagram

A more complete description of the Trucktown architecture and implementation can be found in Andy Tian's 1998 Computer Science Master's thesis.

3.2.1 Expert GUI

There are three expert modules in Trucktown that a user can visit from the main Trucktown map: A garage mechanic, magazine editor, and neighbor. For this version of Trucktown, the entire underlying logic and question dialog is identical with all the experts. The differences between them are the dialog of in the introduction & exit screens, the background images depicting the experts, and the order of the questions.

The interface of the expert advisor is a human that asks a series of questions consisting of multiple choice questions, sliders, and chip allocations. The advisor GUI is as realistic as possible, asking the questions through carefully chosen dialog. The advisor emulates a human being to invoke trust and increase the probability that the advice given is taken by the user.

3.2.2 Expert GUI Functionality

The Trucktown interface contains three different experts. Each expert contains the same basic questions but vary in introductory dialog. These experts ranked highest for trust in the latest GM truck owner survey. The three experts in this version are:

- The Mechanic
- The Editor of Consumer Reports-like magazine
- Neighbor/Friend - Person who has owned many trucks

Figure 4. Experts recommended by the Owl

Appendix C contains the exact wording of the expert questions.

3.2.3 Expert GUI High Level Architecture

A strategy of re-instanciating panels was used to simplify the logic of the expert and aid in the correct implementation of navigational logic. This strategy of not keeping objects forces the panels to not have to "remember" what state every expert panel is in. Creating panels from scratch every time can cause "flashing", a light artifact that flashes across the screen between panels. This visual drawback is not very noticeable and may even be preferred as a source of visual user feedback, causing the user to be aware that they are in a new state.

The state of the user is instead stored in an expert-specific session-history object. It is stored in the SessionModel and is used to reconstruct the state of the user every time a new Panel is instanciated. This abstraction allows for any future change in the user interface.

The different experts were created only after a generic expert was tested. The Expert was copied as three different classes, MechanicExpert, NeighborExpert, and EditorExpert. Each one of these expert classes had within them the same classes found in the hierarchy diagram shown above. The experts are allowed to migrate away from each other in functionality. The users direct responses to each expert are stored in their consecutive "Top Layer Response Database". This expert specific database is translated down to the ResponsesDB (only one instance of this) every time the user enters a new expert module by using the "Direct Response expert".

Expert Questions

The questions posed to the user are shown in a series of screens implemented as separate panels. Each panel has anywhere from one to several questions. When the user goes to the next question panel, any questions that they answer is recorded in the "top layer response database", found in the SessionModel. In this version of the question logic,

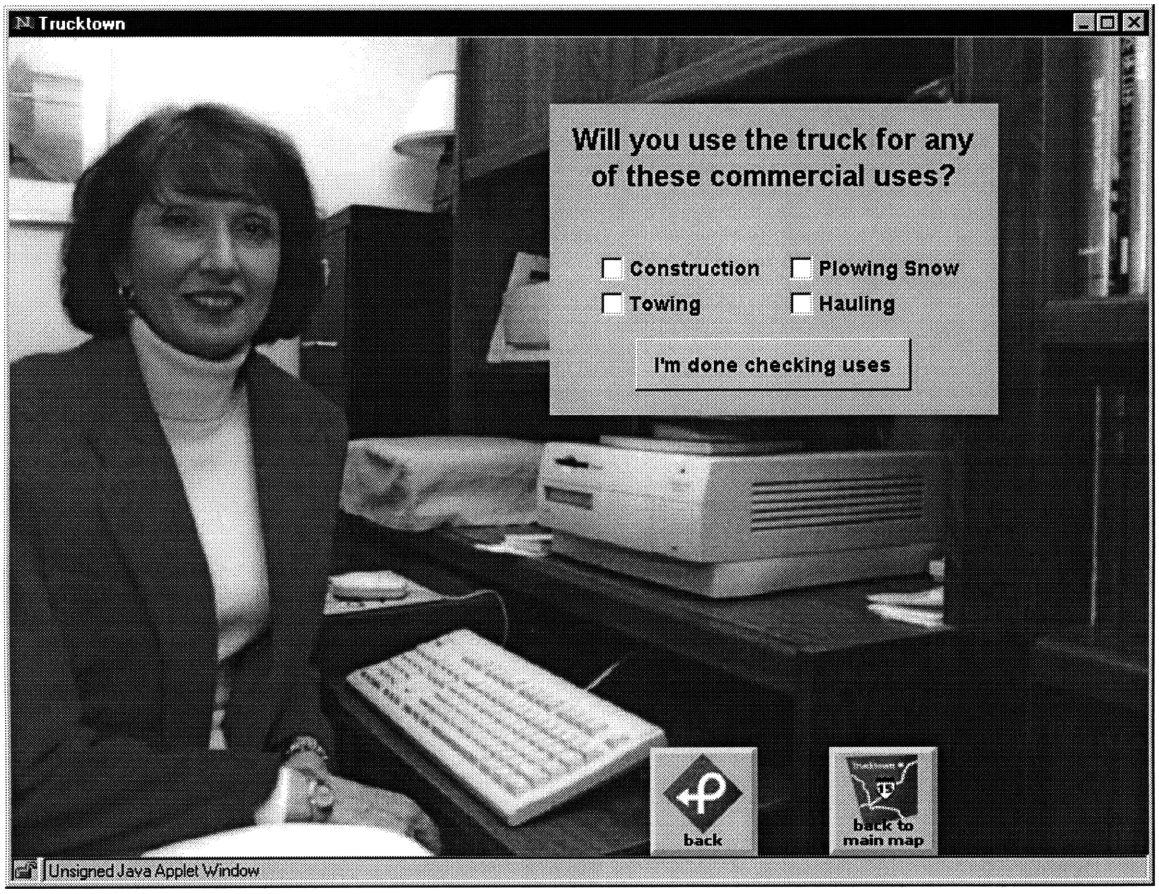


Figure 5. Friendly Human Expert GUI

these panels are shown in the same order regardless of how the user responds to the questions. The user continues through the question panels until the pool of questions is exhausted. The user may press the "previous question" button to flip back though the questions to change how they responded. The user may leave the questionnaire at any time.

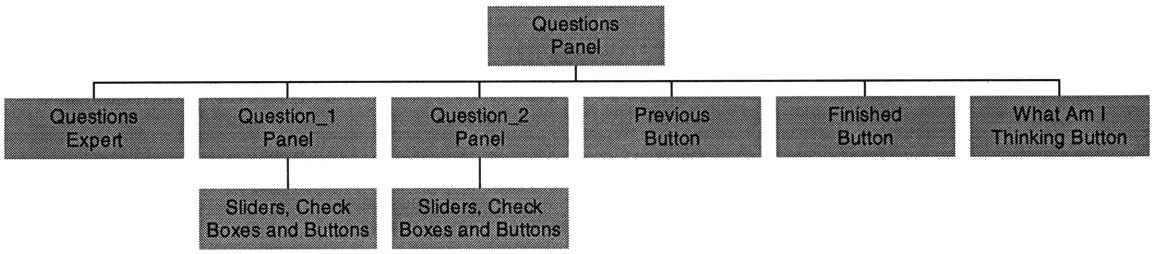


Figure 6. Expert Questionnaire Class Hierarchy

A question expert controls the order in which questions are posed to the user. This expert is currently very simple in that it simply has a preset order for each expert. In future versions of the expert, this functionality can be extended to first ask the questions that will have the most "impact". This will be discussed later in the "Future Extensions" section.

note: In future versions, it is advisable to abstract the content of the experts from the GUI collection engine. Abstracting expert specific information is more flexible than the current implementation. It should not be difficult to replace the current switch routines with calls to an expertGUI object that is passed to the GUI engine.

3.3.1 Expert Logic

The Logic Expert object contains all of the functionality of this knowledge-based system. Its **input** consists of:

1. User responses
2. Truck Recommendation Set
3. Knowledge Databases
    a. Question Correlation Database
    b. Truck Utility Databases
    c. User Historical Response Database
    d. User Market Segment Database
    e. Sales Explanation Database

The logic expert **output** consists of:

1. Ordered recommendation set
2. User specific explanations

3.3.2 Expert Logic Recommendation Functionality

The underlying logic of the needs based expert is abstracted into several experts working in collaboration with each other. These experts are abstracted into a De-Correlation Expert, Utility Analysis Expert, Bayesian Expert, and Market Segmentation Expert. The Consultative Selling Expert will be discussed in a later section.

The de-correlation expert translates the literal responses from the posed questions to a less correlated set of response data. For example, all of the questions regarding off-road driving are compressed into a single response. This data is easier to manipulate by the other experts. The refinement engine is currently a IF/OR rule based system but could easily be extended to implement any number of expert algorithms.

The utility expert analyzes a user's preferences regarding 5 dimensions including price, durability, fuel economy, safety, and horsepower. The information collected about these dimensions is used in conjunction with the dimensional rating of each truck to find the utility that individual has for each truck. The results of this expert are used as the starting a priori probability for the Bayesian expert.

The Bayesian expert logic consists of a probabilistic network that utilizes Bayes Theorem. The essentials of the model is as follows:

1. Each truck has an a priori probability of being bought (or test driven by the user).
2. A relevant question is asked (such as "how many people are in your family?").
3. Previous responses to this question are correlated to trucks people have historically bought.
4. The user's response is used to update the a priori probability of each truck being bought (using Bayes Theorem).
5. The process is repeated for every question the user gave a response for.

The market segmentation expert is rule-based, drawing directly from proprietary GM marketing research. The final segmentation recommendations are ordered using the results from the Bayesian/utility expert.

The elimination expert is a simple filter that eliminates trucks based on user responses. Trucks that are not eliminated fit the users strict preference criteria. A problem with using this algorithm is that the user walks a rule based "landmine" that will be unable to recommend trucks if the user responds with needs that are inconsistent with the available recommendations. For example, this expert would eliminate all vehicles if a user responds that they need a powerful and fuel-efficient vehicle. However, if the user is able to view the underlying elimination as she responds, the user can easily see where the "landmines" are and can react accordingly. This expert is implemented in another module in Trucktown called "AutoMile".

The inconsistency expert is one way to solve the elimination problem that occurs when users attempt to find a recommendation that does not currently exist. An ad hoc implementation of this expert is for it to check for inconsistencies that the user has entered. The expert asks the user if the inconsistent data they responded with is truly what they want to enter, even though it eliminates an unreasonable number of trucks. These questions can be answered when an inconsistency arises or can be saved until after all the questions have been asked. A more general implementation of this expert is to generalize it into a "watch dog" that raises a flag when an unreasonable number of trucks have been eliminated (and when another path would have not created the same problem). This method can be used in conjunction with any expert ranging from the problematic elimination expert to the Bayesian expert. This expert was not implemented, but was designed for future implementation.

3.3.3 Expert Logic Expert High Level Architecture

An analogy to the design of the Logic expert can be described as a room full of various experts with one head mediator speaking for them. The mediator is given all of the information and later emerges with an answer. While coming to a conclusion, the head mediator asks everyone for his advice. The individual experts think about the

recommendation and often consult other experts before responding to the head mediator. The mediator collects all the information and organizes it.

Two general guidelines were used while creating the logic expert architecture:

1. All experts were implemented and encapsulated as separate objects. This encapsulation of AI algorithms allowed for the details of the implementation to be hidden, thus allowing the architecture to be extremely flexible. This also proved to greatly enhance the robustness of the expert system. In practice, this abstraction often occurred at the method level.
2. The system would NOT be optimized by performing partial updates to the recommendation set. The logic expert would re-calculate the top recommendation set between questions with a new response set every time. This eliminated the need to rigorously track the state of incomplete response information. Though computationally intensive, this strategy worked extremely well with current computers (200MHz). Had partial updates been implemented, the expert would have been much less robust.

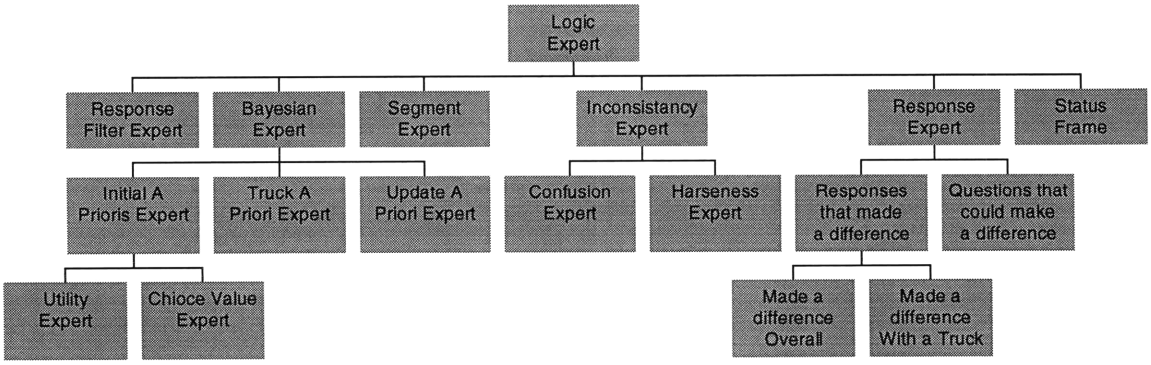


Figure 7. Complete Logic Object Diagram

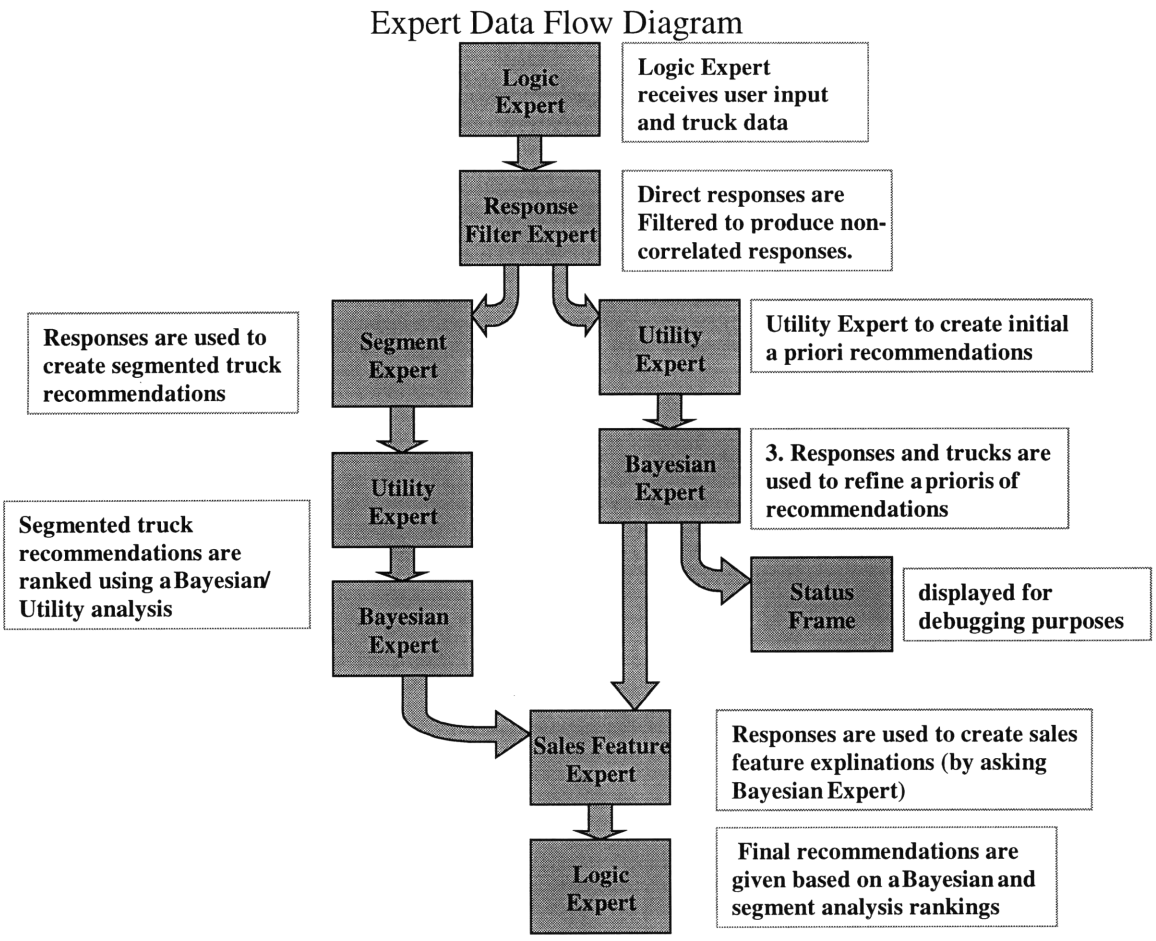


Figure 8. High Level State Diagram

3.4 Bayesian/Utility Expert

The heart of the expert system is the Bayesian/utility expert. The showroom uses it to create the top left and right recommendations. The Bayesian Expert Object does the following:

1. Creates a pseudo a priori probability for each recommended truck using utility analysis.
2. Refines the initial a priori probability using simple Bayesian updating.
3. Ranks the user responses to form user and recommendation specific consultative selling.

3.4.1 Bayesian/Utility Expert Functionality and Algorithms

The Bayesian expert uses historical data to update the a priori probability of each truck. This historical data consists of how previous users answered questions knowing what the "best" truck was for them. The Bayesian expert uses these previous "best" recommendations to infer what is the current best guess.

To populate the conditional probability knowledge database, the marketer simulates 100 users who have bought each truck for a total of 8700 simulated users. For each question, they fill in how truck owners would have answered. This could be done with surveys or historical data, but the marketer can also use their own judgement. For example, 90 people who end up with a large Dodge Ram will say that they like large trucks, while 10 people with a Dodge Ram will instead say they don't like large trucks. The marketer does not have to understand programming to be able to input this data.

The algorithm also incorporates General Motor market data in the form of a utility analysis algorithm.

Utility Analysis Algorithm

The following is the basic utility algorithms devised from past GM studies, Urban/Roberts papers, and industry common knowledge to create the initial a priori probability. Trucks are rated on 5 attributes and the user rates these attributes according to what he is looking for. Trucks that are strongly rated in the attributes the user ranks highly are given high probabilities.

It is not the only way to create this initial probability and is not the dominant factor in the Bayesian recommendation algorithm. Another way to create an initial probability for the Bayesian expert is to rank trucks according to current sales volumes. However, it does provide a very pragmatic way to incorporate utility analysis by using a domain translation function.

Utility Expert Data Flow Diagram

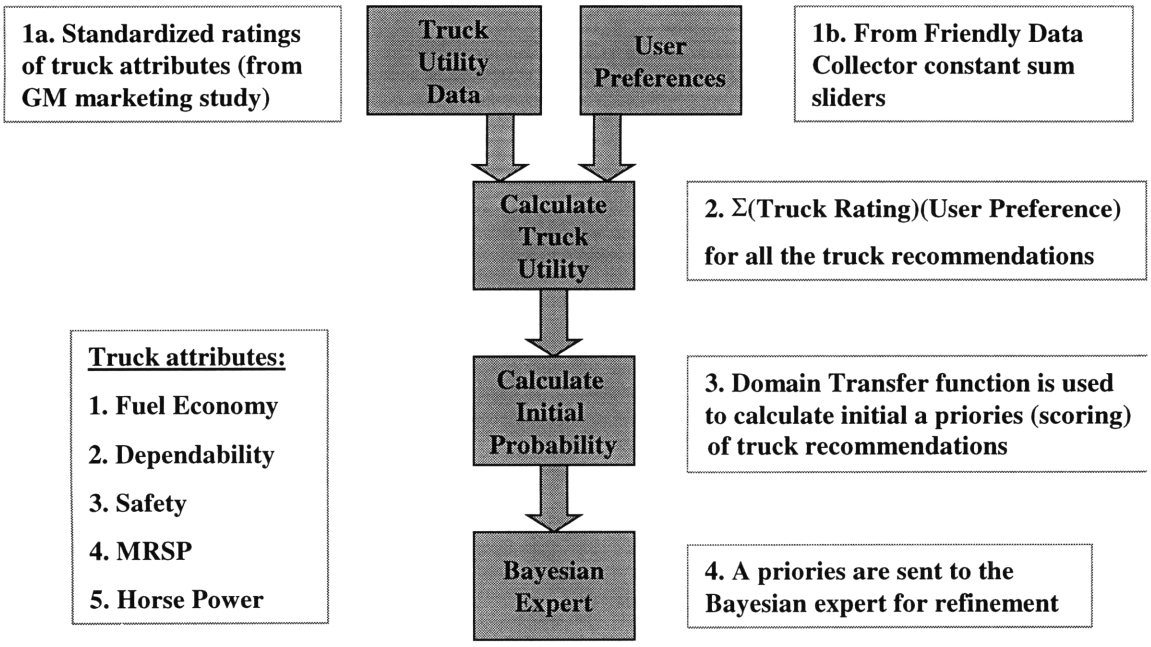


Figure 9. Truck Utility Data Flow Diagram

1. Standardize ratings for truck attributes. (This step is accomplished previous to populating the truck utility database. See **Appendix E**)

$$x_{i,k} = \frac{y_k - \mu}{\sigma}$$

Where

$Y_{i,k}$ = rating for truck k for attribute I before it is standardized.

These attributes are:
Fuel economy
Dependability
Safety
MRSP
Horse power

2. Calculate vehicle utility based upon answers to constant sum sliders.

$$v_k = \Sigma a_i x_{i,k}$$

Where:

a_i = Rating for attribute i (from constant sum sliders)

i = Attributes in constants sums sliders (safety, reliability, fuel economy, power, price)

k = Trucks in database

$x_{i,k}$ = Standardized rating for truck k for attribute i

The user sets constant sum sliders during expert question #7. During usability testing, it was found that users had an extremely difficult time with this method of collection. In future versions of Trucktown, I suggest that the sliders be replaced with a pie chart that the user can manipulate. Instead of the height of the sliders, there would be areas of each piece.

3. Calculate initial probability of purchase.

$$P(K_k) = \frac{1}{(1+e^{-vk})}$$

This domain transfer function is used for its utilitarian qualities. It transforms the truck utility values into probabilities that are found to be a satisfactory approximation of the relative probabilities that the user will buy a truck. The function transforms **V** utility (-infinity, +infinity) to **P** probability (0, 1).

4. **Scale this probability to assume that the user will buy one truck.**

$$P(K_k) = P(K_k) / \Sigma P(K)$$

Before the user reaches the slider question in the expert interview process, the attributes are preset to all be equal to each other. A problem that becomes apparent is that these preset values will give a ranked truck recommendation set, without regard to any user.

Because some truck recommendations will have scored a higher utility than others, the recommendations result with some trucks as having a higher probability of purchase than others. This initial a priori probability was not reviewed by a qualified expert and may not even be realistic. However, this algorithm does seem to push the recommendation set the correct direction and appears to be an excellent first order approximation.

Bayesian Updating Algorithm

The following is the algorithm used to refine the initial probability of purchase. It assumes that the responses are highly **non-correlated.** Making this assumption allows the Bayesian net to be to be greatly simplified by breaking the links between the response nodes.

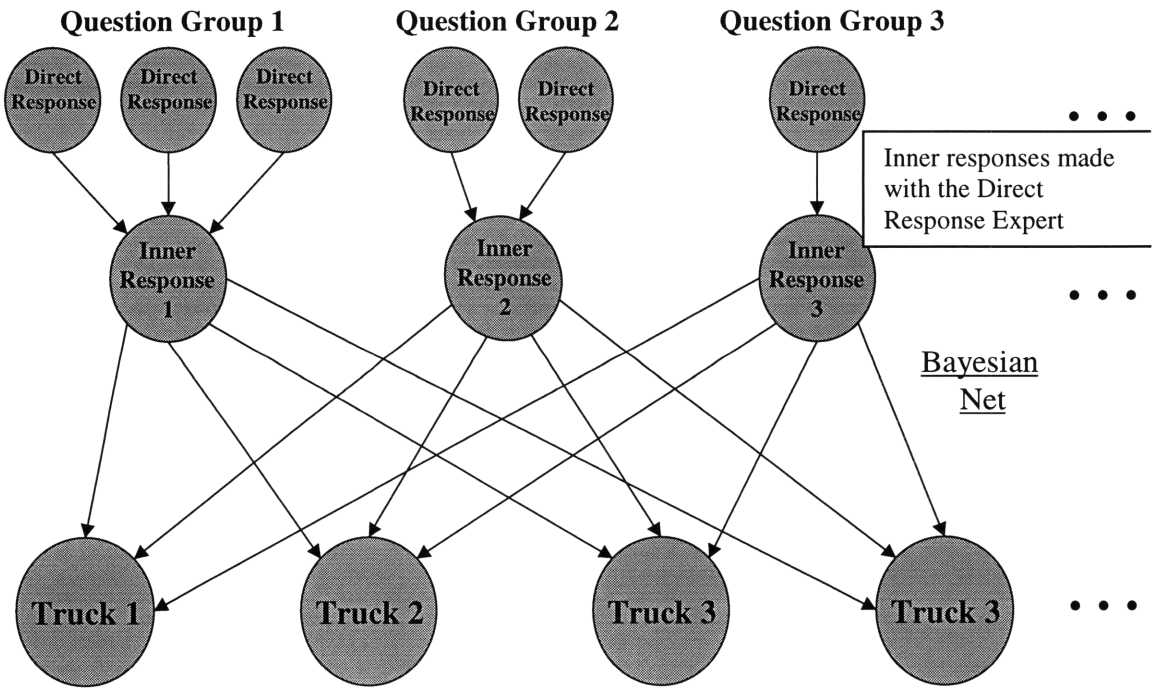


Figure 10. Bayesian Network Node Diagram with direct responses

Although the direct-response expert combines many of the correlated questions together (such as the hauling questions from question screen #3 and #4), the ResponsesDB still has responses that are at least partially correlated. This is solved by populating the historical database with data that is "less harsh", thus canceling out the redundant information. For example, because the engine size questions are correlated, the historical response data is "smoothed out" over those responses, making the effect of each individual engine question smaller, but the accumulative effect at the correct level of influence.

This "fudging" of the data replaces the need of maintaining a more fully networked Bayesian network. The complexity of such a network could be extremely difficult to maintain both in terms of performance and robustness.

Handling the de-correlater is best handled in the direct-response expert. This expert can maintains any number of solutions for combining correlated responses together. It is currently using an ad hoc rule-base in the form of if/AND/OR statements.

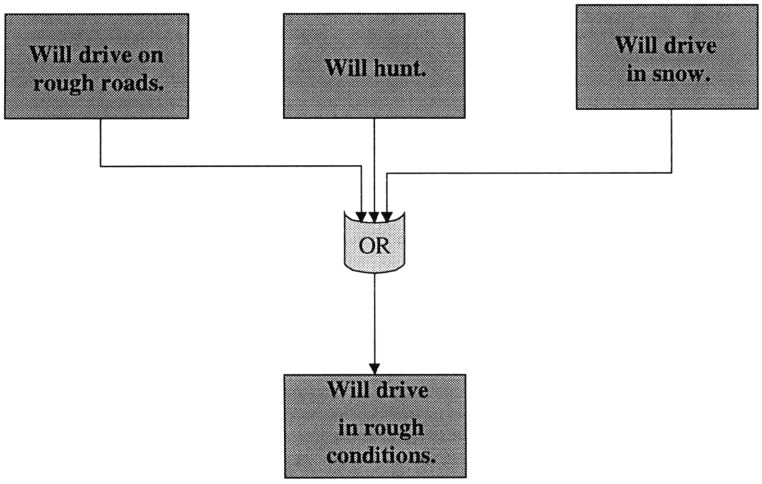


Figure 11. De-Correlation Expert Refining the Direct Responses to the Inner Response Layer of the Bayesian Network

Bayesian Expert Response Update

Bayes Theorem:

$$P(K_k|Q_{q=r}) = \frac{P(K_k)\,P(Q_{q=r}|K_k)}{P(Q_{q=r})}$$

Stuff on the Right:

$P(K_k)$ = a priori probability (originally from the Utility Analysis Expert)

$P(Q_{q=r}|K_k)$ = probability of response to a question knowing they will buy this truck

$P(Q_{q=r})$ = probability of response to a question

Thing on the Left:

$P(K_k|Q_{q=r})$ = probability this truck is the "best" recommendation knowing the user's response to a question

What happens after every update:

$$P(K_k|Q_{q=r}) = \frac{P(K_k)\,P(Q_{q=r}|K_k)}{P(Q_{q=r})}$$

Figure 12. Bayes Theorem Applied

Bayesian Update Example

Biggest/Smallest	Truck = k				Traditional/Futuristic		
	S10	Ranger	Ram			S10	Ranger
$P(Q_{q=1}\|A_{a=})$	0.70	0.80	0.00		$P(Q_{r=1}\|A_{a=k})$	0.70	0.50
$P(Q_{q=2}\|A_{a=})$	0.27	0.10	0.00	From past customers Conditional Probabilities	$P(Q_{r=2}\|A_{a=k})$	0.27	0.20
$P(Q_{q=3}\|A_{a=})$	0.02	0.06	0.01		$P(Q_{r=3}\|A_{a=k})$	0.02	0.15
$P(Q_{q=4}\|A_{a=})$	0.01	0.03	0.07		$P(Q_{r=4}\|A_{a=k})$	0.01	0.09
$P(Q_{q=5}\|A_{a=})$	0.00	0.01	0.93		$P(Q_{r=5}\|A_{a=k})$	0.00	0.06
	1.00	1.00	1.00				
$P(Q_{q=1})$	0.67				$P(Q_{r=1})$	0.62	
$P(Q_{q=2})$	0.14				$P(Q_{r=2})$	0.24	
$P(Q_{q=3})$	0.04			Joint Probabilities	$P(Q_{r=3})$	0.07	
$P(Q_{q=4})$	0.03				$P(Q_{r=4})$	0.04	
$P(Q_{q=5})$	0.12				$P(Q_{r=5})$	0.02	
$P(A_{a=k})$	0.33	0.55	0.12	A Priori Probabilities	$P(A_{a=k})$	**0.62**	**0.38**
$P(A_{a=k}\|Q_{q=})$	0.344	0.656	0.000		$P(A_{a=k}\|Q_{r=})$	0.694	0.306
$P(A_{a=k}\|Q_{q=})$	**0.618**	**0.382**	**0.000**		**$P(A_{a=k}\|Q_{r=})$**	**0.686**	**0.314**
$P(A_{a=k}\|Q_{q=})$	0.162	0.809	0.029		$P(A_{a=k}\|Q_{r=})$	0.178	0.822
$P(A_{a=k}\|Q_{q=})$	0.120	0.598	0.283		$P(A_{a=k}\|Q_{r=})$	0.153	0.846
$P(A_{a=k}\|Q_{q=})$	0.000	0.047	0.953		$P(A_{a=k}\|Q_{r=})$	0.000	0.960

Figure 13. Bayesian Update Example

1. **General probabilities are generated from user response historical database.**

$\mathbf{P(Q_{q=r})}$ = The probability response **r** to question **q** (generated from historical response database or judgment probability of **r** response given prefer $\mathbf{K_k}$)

$\mathbf{P(Q_{q=r}|K_k)}$ = The probability response **r** to question **q** given the user will purchase truck **k** (generated from historical response database or judgment probability of **r** response given preference $\mathbf{K_k}$)

These probabilities are calculated every user session to allow the historical database to be more easily updated. The probabilities are generated for all responses and are used to in the Bayes Theorem.

2. **Iterate through a Bayesian update to recalculate the probability of purchase for all responses.**

We can calculate the probability of selecting truck **k** given the user answers **r** to question **Q** $\mathbf{P(K_k|Q_{q=r})}$ using Bayes Theorem.

$$\mathbf{P(K_k|Q_{q=r})} = \frac{\mathbf{P(K_k)\,P(Q_{q=r}|K_k)}}{\mathbf{P(Q_{q=r})}}$$

$\mathbf{P(K_k)}$ = The a priori probability of purchasing vehicle k (originally from probabilities generated by the constant sum slider)

3. **After every iteration through the Bayesian update, replace the $\mathbf{P(K_k)}$ with the newly calculated $\mathbf{P(K_k|Q_{q=r})}$.**

This substitution reflects the logic that the new a priori probability for the recommendation now incorporates the knowledge from the previous responses.

4. Sort truck recommendations by $P(K_k|Q_{q=r})$.

The top recommendations are filled with sales features and passed back to the Logic Expert. Sales features and sales problems will be discussed later in the "Consultative selling" section.

Improving recommendation quality using the historical database

Over time, the historical database can be refined using the user sessions with the expert. This will allow for more accurate estimations of $P(Q_{q=r})$ and $P(Q_{q=r}|K_k)$. The difficult and subjective part of this is finding **K** the "correct" or best truck recommendation to correlate with **Q** that user's response information.

This best truck recommendation (the truck that is assumed to be the best truck for that user) can be recognized by some action of the user. What could be a "good" recommendation?

1. Trucks the user schedules a test drive with.
2. Trucks that the user clicks on to get to the truck panel.
3. Trucks that the user collects a lot of information on a recommended truck.
4. Trucks that the user specifies in a post-online survey as ones they felt were good expert recommendations.
5. The user could be asked in the showroom if the truck they are being shown is a good recommendation.
6. Trucks that the users buy (not a feasible solution in this research project).

Several of these user actions could be combined in a comprehensive strategy to create a first order approximation of increasing the recommendation quality. Different strategies could be quantitatively compared if the recommendation quality is tracked and bench-marked (discussed in the "Future Extensions" section).

Truck utility data could also be updated to better represent "true" attribute rankings. This would be a user-oriented way of updating. It may not be philosophically correct to use this approach if the utility data is meant to represent the some utilitarian expert view of the recommendation set and is not meant to represent how the user excepts the results. This view of the recommendation may produce better recommendation quality, but ironically at the expense of utilizing the expert as a sales mechanism.

3.5 Segment Expert

The segment expert is a rule-based algorithm that was developed by Frank Days using proprietary market research from General Motors. A segment refers to a category of buyers that are looking of a particular type of truck. There are 8 segments, each segment is assigned 4 truck recommendations. Before responding to questions, a user begins belonging to all of the segments. Segments are eliminated as the user responds to the expert questions, until they end up in exactly one segment.

The remaining segment trucks are ordered ranked using the Bayesian/Utility Expert before they are given back to the Logic Expert. The Showroom shows the top ranked truck recommendation as the bottom left recommendation and the "What I'm thinking panel" shows the two top ranked segment trucks as the bottom left and right recommendations.

3.6.1 Integration with the Showroom and "What I'm thinking Panel"

Both the Showroom and the "What I'm thinking Panel" are used to show the user the top expert recommendations. The Showroom is a completely separate module in Trucktown and has relatively deep functionality to explore the specifications of each recommendation. The "What I'm thinking Panel" is a sub-panel in the expert module and

its purpose is to convey the expert logic in a simple form – thus eliminating the "black box" effect in version 1.0 of the expert.

3.6.2 Showroom High Level Functionality and Architecture

Both the Showroom and "What I'm thinking Panel" are fairly similar in high level user interface in that they both show the user the top recommendations in a truck icon format. The underlying technical structure is similar in that they both get recommendations by creating a Logic Expert and having it evaluate the recommendation set and the user response database.

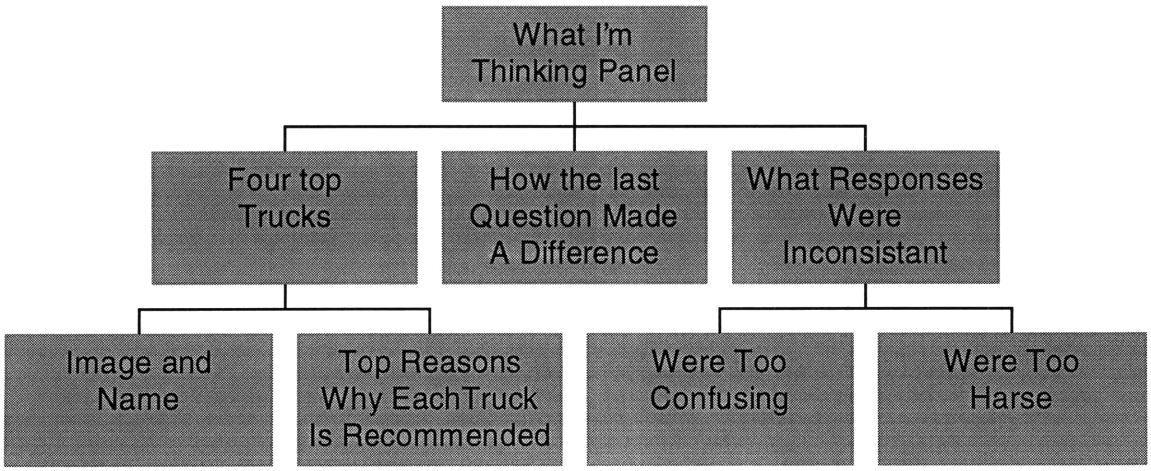


Figure 14. "What I'm Thinking Panel" Object Diagram

The recommendations are shown to user as four truck images with accompanying labels. These trucks are the four trucks that are returned by the Logic Expert using the public method "get4RecommendedBayesianAndSegmentTrucks".

The only exception to the recommendation set is when the user has told an expert the last truck the user has bought. In this case, the bottom right truck in the Showroom(which was the second ranked segment recommendation) is replaced with the truck previously owned by the user. The user is told that this truck is put there to be able to be compared to the other recommendations. The actual reason is that users who have just bought a truck will react negatively to an expert who does not give them their truck. This

functionality causes the post-purchasing user to trust the expert more even if their new truck does not come highly recommended.

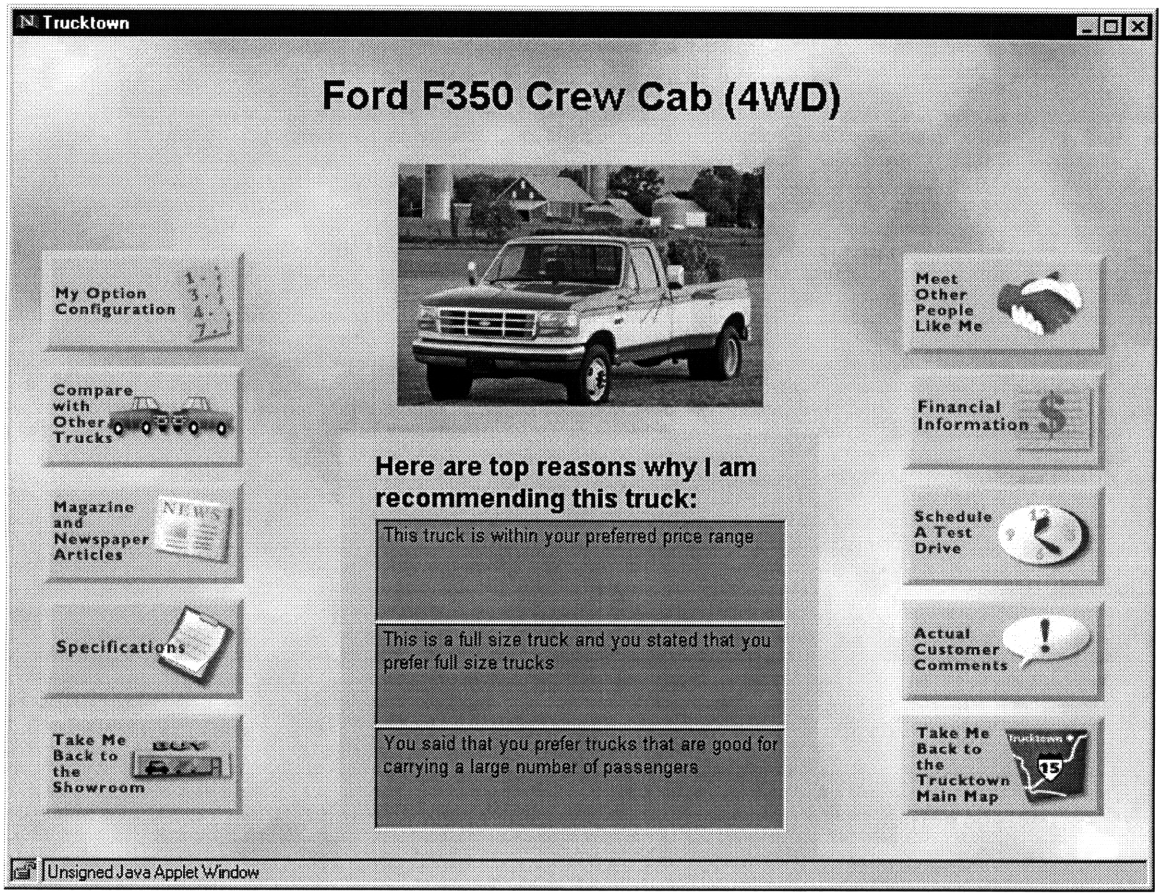


Figure 15. Showroom GUI

4. Consultative Selling Expert - Using Response Data in the Sale of the Recommendation

The response information is used to generate a tailored consultative sales explanation for each individual user. The same truck may be recommended to different users but for very different reasons. Instead of giving a generic explanation for a recommendation, the sale can be targeted directly at the user with their particular needs in mind. This can be important especially if the user does not precisely fit a segment. For example, a user may be given a recommendation that does not exactly fit their needs. The user will trust the

recommendation more if an explanation is offered as to why it is the "best fit" considering what is available.

4.1 Consultative Selling Functionality

Creating a sales explanation

1. Pull sales features and problems out of the sales database according to how the user responded.

User Responses	"impact"	sales feature	sales problem
Wants a big truck	?	This truck is BIG and STUDLY	This truck is TOO SMALL
Wants to go offroading	?	This truck is good for off-roading	This truck is not made for off-roading
Wants a cheap truck	?	This truck is in your price range	This truck is out of your price range

2. For the recommended truck, figure out the "**impact**" that each response had on the recommendation.

User Responses	"impact"	sales feature	sales problem
Wants a big truck	"+ 3%"	This truck is BIG and STUDLY	-
Wants to go offroading	"+ 2%"	This truck is good for off-roading	-
Wants a cheap truck	"- 1.5%"	-	This truck is out of your price range

3. Generate a sales explination based on the remaining sales features and pr

This GMC is BIG and STUDLY and is good for off-roading. However, I should point out to you that this truck is out of your price range.

Figure 16. Creating a Sales Explanation

The basis of the consultative selling expert is as follows: There are features and problems corresponding to every user response. All truck recommendations have either a feature or problem according to how the response "impacts" the recommendation's a priori probability of being a "good" recommendation. This "impact" can be measured and is used to rank the features and problems. The top features and possibly the top sales problems are then used in the consultative sell.

Using too many sales features may overwhelm the user with too much information. Too many reasons given may create the impression of chaos, causing him or her to feel that the expert did not have a specific strategy in making its recommendation. Though it is good for the user to have access to all of the reasons why a recommendation is suitable, it is preferable to point out one to three as the primary reason for making the recommendation. The user will anchor to these reasons and come back to it when he or she becomes overwhelmed with information. One simple reason will not confuse the user, whereas many good reasons will make the user's decision process more complex.

4.2 Consultative selling Algorithms and Architectural Details

The consultative selling expert finds which user responses had the most impact in making its recommendation, and weighs this with how good of a sales point it is. The algorithm for finding sales features is as follows:

1. Filter out responses not made by the user.
2. Calculate "impact" of each response on final recommendation
    3. Calculate the a priori probability of a truck with no questions answered
    4. Calculate the a priori probability of a truck with the one response
    5. The "impact" is the difference between 3. and 4.
6. Weight "impact" score with "sales point importance factor"
7. Rank "impact" scores
8. Correlate "impact" scores sales features and problems
9. Filter out "impact" scores that are close to zero (indicating weakness)
10. Generate user dialog sales features and problems

"Impact" is calculated by feeding the responsesDB to the Bayesian Expert once empty and once with the response included. Another way to measure impact could have been to compare a full response set with a full response set minus the response in question. However, measuring impact **in context** of other responses was found to be very problematic due to the a priori probability being so distorted. The "impact" of responses standing by themselves was found to be much more accurate representation of the relative importance of these responses.

This Bayesian calculation could be generalized to the Logic Expert level if other heuristic expert algorithms are implemented. Instead of deriving "impact" from changes in a priori probability, it can be found from differences in other scoring schemes.

All trucks have a sales feature and problem importance factor for each response. This factor is needed in situations where the "impact" of the sales feature or problem does not reflect what the marketer wants to describe to the user. For example, if a recommendation was a GMC, and user did not check that they were looking for a Toyota, this response may have impact on the final recommendation but is awkward to use in a consultative selling. The factor allows the marketer to change the relative rankings of features and problems.

The sales point importance factor can also be dynamically altered based on if the system learns that the recommendation was acted upon- implying that the consultative selling was affective and should be used more often. This is very similar to the dynamic updating of the recommendation logic except that the same user actions are used as evidence that the consultative selling was the cause of the sale.

5. Java Implementation Issues

Java is used in the implementation of Trucktown because it is a common standard for Internet based programs and is excellent for developing robust logical systems. Its drawback is that Java is still immature to use in implementing a graphical user environment. Java 1.1 was chosen over to Java 1.0 to allow for the more rapid development, even though it limits the choice of browser and operating system platform. The object oriented abstracted design and client-server architecture of Trucktown has allowed it to be very extendible and flexible.

5.1 Implementing the Expert Logic

Java is an excellent language for rapid development of complex logic. Its simplicity makes it very robust and easy to manipulate. Java is a de facto standard of the Internet, making it a well understood environment for professional programmers to implement the client-server architecture needed for dynamic updating of the logic, user-history data collection, and implementation of server-side, computationally intensive expert algorithms. Because commercial Web browsers generally support the Java environment, the code-base of this research implementation of Trucktown can be extended into a commercially viable product.

5.2 Implementing the GUI

The main drawback of a Java-Internet implementation of Trucktown is its immaturity as a graphical interface. Bandwidth is an issue and there are no mature tools to quickly implement the user navigation. Because of these problems, the logic of the expert is completely abstracted from the user interface in anticipation of the future development volatility.

5.3 1.0 versus 1.1

The previous version of the Trucktown environment was implemented in Java 1.0. The main change in programming tools was the new use of Java 1.1. Though not necessary in implementing correct functionality, the more developed version of Java allows for much faster implementation, allowing for approximately 20%-30% more functionality developed within the same schedule. The tradeoff is that the user is then limited to using Netscape 4.05. In a controlled research environment, the tradeoff for added functionality is acceptable, but will make the migration to commercial product more time consuming-particularly if the user interface is reused.

The event modals of Java 1.0 and Java 1.1 was not used for class communication to avoid a restructuring of the Trucktown communication modal if the technical course of the project were to change. Instead, all communication was made by public or protected methods. Data is either passed explicitly by reference or through the class SessionModel. In terms of extendibility, this methodology is not optimal. The structure the component hierarchy in this methodology turned out to be quite rigid with respect to method calls. Objects depended on the type of object that instanciate them, causing problems with reuse. This methodology turned to work out well when dealing with computational objects (such as the experts) but caused problems with respect to the GUI and navigational structures. Most notably, the migration of GUI sub-panels from the Showroom to the Automile proved to cut across the method-calling architecture and proved to be very time consuming.

5.4 Object Oriented Methodology

The main change in methodology from the previous version was fuller abstraction in the Java environment with more complete use of object-oriented design. These changes in methodology were necessary to allow for the development of a more complex environment. An attempt was made in the first version to manipulate all the components and screens in one "master" environment. This proved to become unwieldy, making a slow user interface and an architecture that could not be extended. Because of the master environment, it was impossible to partition tasks to separate programmers. This was the main cause of difficulty in the summer of 1997 when new junior programmers came into the project and attempted to extend the architecture. In this version of Trucktown, the architecture of the program was broken into a hierarchical structure that made use of abstract representation. For example, the expert interaction with the user is abstracted into its own module that can be programmed as a separate task without the programmer having to "know" everything about the rest of Trucktown. The expert module is broken down into its user interface and underlying logic. The underlying logic is broken up into distinct experts, such as the Bayesian expert and the market segmentation expert. The

experts work with Truck and user response "objects" and pass them around to communicate. The architecture allows for such extensions as a new expert (such as a neural network) to be implemented by another programmer or the user interface to be quickly replaced with another implementation in even another language.

5.5 Client-Server Architecture

The other major addition to Trucktown is the very robust client-server architecture. The most immediate impact of this architecture is that it allowed the use of open database connectivity to Access (or any other database). This allows for an immediate and

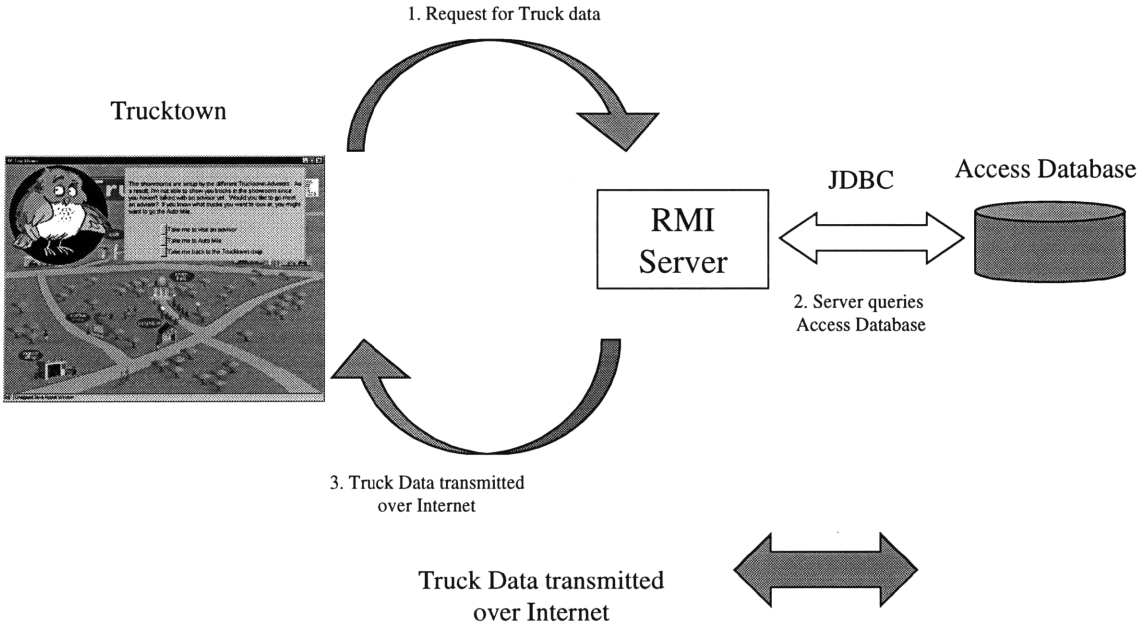


Figure 17. Client-Server Architecture

complete integration with content that could be changed and updated by a non-programmer. This proved very important to this version of Trucktown and would be absolutely critical for a commercial implantation of any heuristic configuration engine. This client-server architecture allows for computationally intensive AI algorithms to be run on a faster server machine and allows for more complex data mining requests that

could be used by expert algorithms. See Andy Tian's 1998 Thesis for a fuller description of this architecture.

5.6 Java Conclusion

Overall, Java have proved to be extremely usable and extendible regarding the back-end logic and still relatively immature as a front-end environment. Version 2.0 of Trucktown was more object-oriented and had a more full client-server architecture than version 1.0. This proved very successful in allowing several different developers to work on the same project together. It has very good abstraction and promises to be very usable in the next version of Trucktown.

6. Extensions to this Research

6.1 Extensions Introduction

There are four ways the expert system research can be extended. These extensions include improving the recommendation quality, use in new product development, improving the GUI, generalizing the engine to sell other products, and further integration of the recommendation engine with the sales process.

The recommendation quality can be greatly improved with more sophisticated modeling techniques and more rigorous testing methods. The system could be made better by handling correlated questions, handling responses inconsistent with current possible recommendations, dynamically ordering questions, building other major expert algorithms, and incorporate more dynamic learning.

This automated sales interface can be made into an excellent marketing tool. User interaction could be systematically captured and analyzed to see what configuration users are attempting to create.

The GUI could be improved to increase usability and trust. The expert logic is abstracted from the GUI and several development options are available. These options are discussed in 6.4 Java Extensions.

The expert engine could be generalized to sell other products. The "knowledge" of the system is almost completely captured in the form of databases. Filling the database with camera information could instantly create a camera salesman.

The recommendation engine can be further integrated with the sales process in 2 ways: 1. Further customization of the sales process based on user interaction and responses (such as the "Consultative Selling Expert"), and 2. Integration with underlying order configuration software or other sales-force automation systems (such those of Trilogy, CWC or Siebel).

This section is broken into three parts: 1. Direct extensions of this research 2. Major extensions of this research and 3. Technical Java extensions that may occur in the near future of this research. There is a lot of ideas and many of them may not be worth pursuing due to funding and time constraints. This section does not cover the major trust issues the MIT Sloan Electronic Commerce Project has been researching and instead concentrates on the expert system research.

6.2 Direct Extensions of this Research

Before suggesting global directions this project can be taken, I will first discuss several possibilities that leverage the existing Trucktown architecture and would quickly yield interesting results. These extensions can use the expert knowledge base, graphical

interface, and general expert architecture described in section 3, Logic Expert. These extensions include generalizing the Logic expert to use more algorithms, using the expert to generate marketing data, and creating a general ordering algorithm that would allow the expert to converge more quickly on a narrow recommendation set.

6.2.1 Posed Question Grouping Translated to Non-Correlated Responses

In dealing with the responses from the user, we find ourselves with a paradoxically "non-correlated but inconsistent" problem. We use AI algorithms that assume that the questions are independent of one another, and yet we profess that the user can answer in an inconsistent way. Unfortunately, if questions are completely independent from one another, any question will not indicate the probability of how a user might answer any other question. For this reason, we can group correlated questions that could be responded to in inconsistent ways and treat them as one question in the underlying logic. We begin to deal with this problem by creating a "Direct Response Expert" that translates groups of correlated responses to single responses that the rest of the logic expert uses (particularly the Bayesian expert).

A salesman (or any expert) has a general process by which they collect and process information from a user. To make a recommendation, the expert tries to answer a set of underlying questions. These basic underlying questions may be very difficult to get at by directly asking the user. For example, an underlying question could be, "Is a customer willing to spend more in order to get the extra bells and whistles of a product?" This could be a very difficult question to get at directly. The person doesn't know why they may want the extra features. They may answer yes to this question but in fact be price-insensitive. So the expert will pose several correlated questions to try to "get at" this underlying question. For the above question, the expert could pose questions to find out if the customer would receive a lot of utility from features. "Does the person want to use the product as a symbol of success?", "Does the person's previous product have these bell and whistle features?", "Is the person going to use the product simply for the basic

features?", "Is the person a serious user of the product, or a casual user?", "Regardless of price, does the person value the bells and whistles?" The expert will continue to pose these questions until they are sufficiently confident that they can correctly answer the underlying question.

After the expert finishes collecting information to answer these underlying questions, he shifts gears and thinks about the recommendation that he will make. The expert "throws away" the information they collected in the form of posed questions. He makes a recommendation based on the information found by analyzing the underlying "mental" questions. These underlying questions are simpler and more non-correlated to one another (making every question "count"). The underlying questions fit well with the logic making the recommendations. Because of the buffer of these posed questions, the underlying questions have a much higher probability of being answered correctly due to the fact that the same question was posed several different ways and the user has answered them in a consistent fashion. If the user does not answer these posed questions in a consistent manner, an "Inconsistency Expert" has a chance to let the user correct themselves or make sense of what the inconsistent user really means. The next section discusses how this expert work could work.

Aside from ordering questions to minimize the length of time it takes to confidently arrive at a recommendation (see 6.2.3 Dynamically Ordered Questions), questions can be clustered into "question subjects"- several highly correlated questions scripted to understand the same underlying subject. For example, if the expert logic wants to clearly "know" what a person's price range for buying a truck is, essentially the same question can be asked in several different ways to get at same subject response. For example, several questions could be asked to find how expensive a recommended truck should be:

1. What is your price range for a truck?
2. How much could you afford each month in car payments?
3. How much could you afford as a down payment?
4. What is your disposable income per year?
5. How much is your rent or house payment?
6. Will your employment reimburse you for the cost of owning this truck?

7. Would you be willing to pay extra to have all the bells and whistles?

Grouping questions in this way prove extremely useful in helping solve the "non-correlated but inconsistent" problem. All of these questions are used to get to the same underlying question- how expensive should the recommended truck be? These high level questions that are posed to the user can be correlated and then translated to answer the underlying question. This underlying question is then used in the recommendation logic.

This translation from "posed question" to "underlying question" can be done in several ways. As a first cut, one question within a question subject is asked. Though this forces the user to be consistent by virtue of not being able to contradict themselves, it allows the highest probability that this underlying question was answered incorrectly but at least will converge on an answer very quickly. This philosophy is acceptable as any answer (accurate or not) is better than no recommendation at all. It also closely models how a real expert behaves.

There are two questions the expert can ask:

1. Questions in a new "Question Subject"- Getting as many of the underlying questions answered as possible before asking correlated questions (that may be responded to inconsistently). The problem with this tactic is that if the user leaves early, the recommendation may be wrong due to questions being answered incorrectly.
2. Questions in "Question Subjects" that already have an answer- Ask all questions in a question group to become confident in a response. The problem with this tactic is that the user leaves early, the recommendation may not have enough information to be very confident about its overall recommendation, thus possibly giving a wrong recommendation by virtue of having incomplete information.

De-Correlation Algorithm

Every underlying question can have three states:

1. not answered
2. answered
3. answered with confidence

It could even be the case that an underlying question could start out as being answered by virtue of having a high probability response (making an un-confused but possibly inaccurate judgment).

As a compromise, the strategy of answering as many underlying questions as possible until a sufficiently narrow range of recommendations is found (such as 30% probability that one of the top four recommendations is correct). After this confidence level in the recommendation is achieved, question groups are revisited to bring up the level of confidence in their underlying questions (due to their inconsistency expert finding that the question is answered with confidence).

A new question group is started as soon as these question groups are answered with confidence. The rest of the question groups are tackled as soon as the last one is tackled.

This grouping structure is excellent in its abstraction to allow other expert algorithms such as utility analysis, neural nets other algorithms that don't assume independence. Each Subject Group could be handled on a case by case basis. Currently, a rudimentary subject grouping is used in "TopLayerResponseDB" for correlated questions such as hauling and off-road driving. These are handled using a simple if/or rule-base (see **appendix A**).

Another more general strategy of dynamically ordering questions can be employed to decide what to ask next, and will be discussed in the next few sections.

6.2.2 Inconsistency Expert

Users may answer questions in a way that will narrow the recommendation set incorrectly or in a way that will confuse the expert. This suggests one of two actions:

1. The user needs clarification regarding how their responses may be incorrectly effecting the expert recommendations.
2. The marketer needs the feedback regarding how peoples' true preferences are not being serviced in the current recommendation set.

This expert is extremely useful when users are attempting to configure a recommendation that does not exist. For example, a user may be attempting to find a truck that is both small and can tow a large boat. Because this recommendation does not exist, the expert automatically shuffles the user to the small truck recommendation because it is most likely what the user is looking for. This expert points out to the user what the trade-offs are and collects this information for the marketing department for future product development.

Algorithm to Calculate Inconsistencies between Two Responses

1. Calculate the a priori "baseline" of no questions answered for all recommendations
2. For each response pair
    3. Add truck a priori difference between responses when in opposite directions of "baseline"
    4. Subtract truck a priori differences between both responses and "baseline" when in same direction from "baseline"
5. Create an nXn matrix of the "inconsistency scores" between answers.
6. Summing the nXn matrix will give a score of how inconsistent overall the user is.

The "inconsistency scores" in #5 of the above algorithm could be used to determine when to ask the user if they wish to change their response. However, this may not yield good results because the alternative responses may be no better. A response may also be

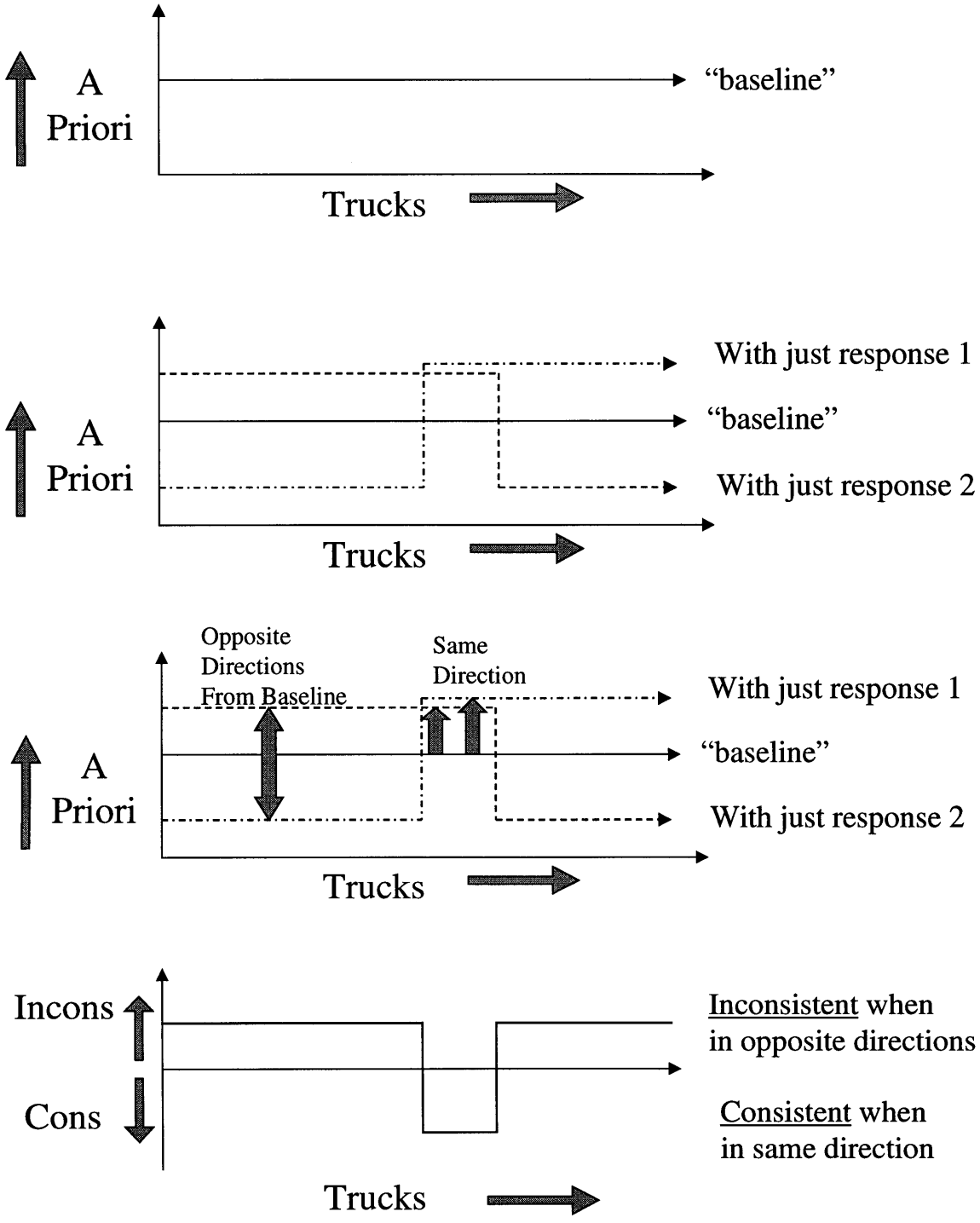


Figure 18. Visual Representation of Inconsistency Algorithm

inconsistent with several other responses and any one of its inconsistency scores may not be very high.

The value in #6 can be used in "what if" scenarios to determine what changes can be suggested to the user that will minimize the inconsistency in the response set. Minimizing inconsistencies is similar to the strategy of dynamically ordering questions. The inconsistency expert would play "what if" scenarios by determining what questions in the past could be changed to yield a response set that is more consistent with current truck recommendations.

Note that this algorithm will give a high score to response pairs that will separately bring the expert to opposing conclusions. This is called an elimination inconsistency. It is "bad" because it causes the expert to eliminate recommendations that may in fact be "good" recommendations.

The combination of responses could cause more "confusion" than if each response was analyzed separately. Another way to define inconsistency is to look for response pairs that separately bring the expert to a narrower decision (i.e. a recommendation set that has a higher variance) than the combined pair. This is called a confusion inconsistency.

This nXn matrix of inconsistency scores can be pre-made by setting all the other responses to unanswered. Because the questions are multiple choice, there is an n^4 matrix of inconsistency information. Utilization of this method would be faster at run time at the expense of the space required by a hyper-cube of information.

Inconsistency Expert as a Recommendation Tool

A problem with needs based configuration is that people are often inconsistent with their espoused needs. A user who responds in an inconsistent way has a lower chance of receiving a recommendation that is "right" for them due to the expert using response data

that does not fit well or correctly with any of the recommendations. For this reason, it is advantageous to help the user be more consistent in their responses by bringing to their attention questions that they have answered inconsistently with the truck recommendations that are currently available.

The method used to find possible inconsistencies in the user's logic is independent from the type of basic expert that is used. The fundamental mechanism the inconsistency expert uses to determine inconsistencies are derived from expert "impact" described in the "Consultative selling" section.

An inconsistent pair of questions are those that by themselves do not bring the expert to the same conclusions. This makes it difficult for the expert to be sure that the user is clear about his or her response. This may cause the expert to be too sure of its recommendation (by eliminating "good" recommendations) or they may cause the expert to become confused and not get any closer to any conclusion. These two types of inconsistencies are called **elimination** and **confusion inconsistency pairs**.

Upon finding egregious inconsistencies, the expert can tell the user that there may be a problem with their response. Generic dialog could appear as follows:

> "You said answer **A** to question **1**, but said answer **C** to question **2**.
> This may eliminate recommendations that you may actually want. Are
> you sure that this is how you want to respond to these questions?"

The user is given a chance to change their responses. If they do change their responses, a more consistent user response set will be created.

Inconsistency Expert as a Market Research Tool

Market researchers can use an inconsistency expert to find:

1. Characterizations of recommendations that the user may be seeking but are not available.
2. Questions that are ambiguous to the user.

A complete recommendation set would contain a custom made recommendation for all users. For example, a five question true or false questionnaire could yield up to 5^2 or 25 possible unique recommendations the user can be looking for. In practice, the actual recommendation set supports the recommendations that have the highest probability of being asked for, generates the most revenue, or is what was chosen at random. Collecting the recommendations that the user is "looking for" may show gaps in the marketplace that are currently not being serviced. Combined with a production utility table (such as profit margins on product recommendations), it would provide marketers with a systematic methodology to create new products.

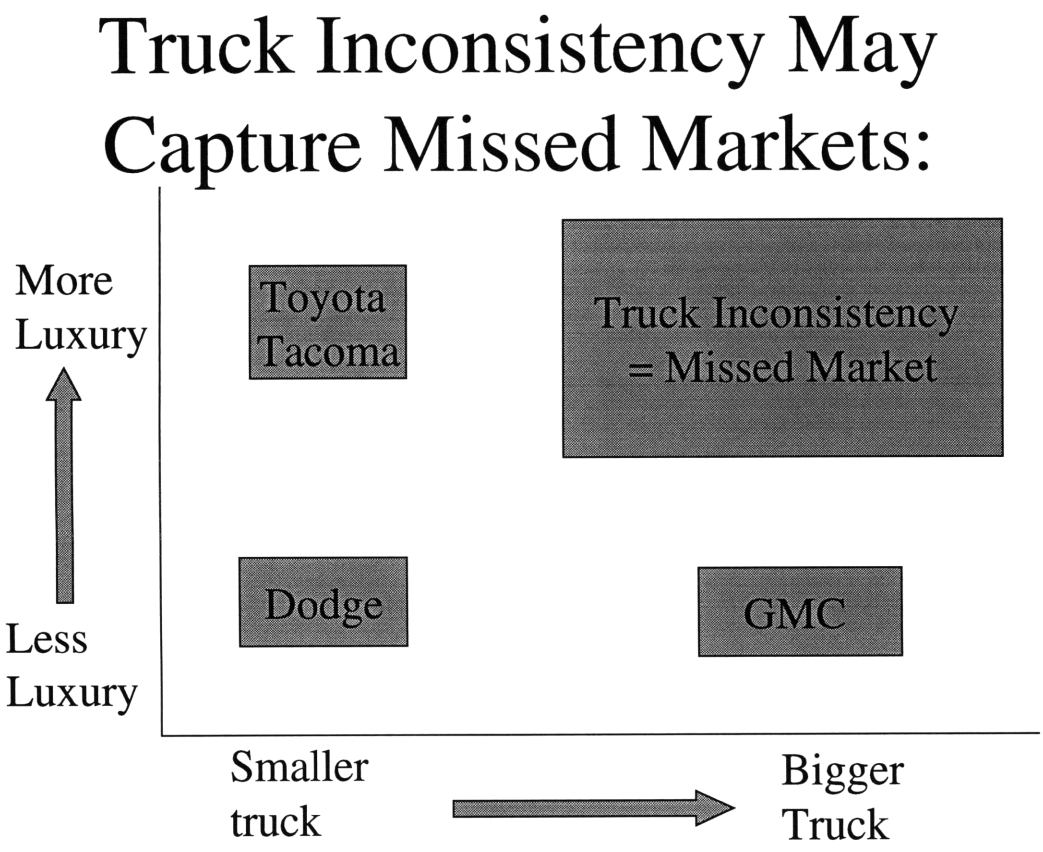


Figure 19. Fictitious Example of a Market Hole

For example, if a user is searching for a large, powerful truck but with a plush and luxurious interior, an inconsistency expert may warn the user that there are currently no

configurations of this type, and suggest an alternative. This expert would also collect data on this configuration that users are repeatedly trying to get. This market hole is then brought to the attention of the marketing department, who can then decide if they want to service this un-serviced market.

Confusion inconsistencies generally do not "hurt" the final recommendation in the same way that elimination inconsistencies do, thus making them less important in generating a "better" current recommendation. They just neutralize the impact each question has on the final recommendation. However, they are just as important as elimination inconsistencies in finding market holes and both types should be captured if the inconsistency expert is being used to find market holes.

Inconsistent response pairs can also be correlated to other inconsistent response pairs- meaning the data can be mined to answer the question, "Of the user group that answered a question inconsistently, what other questions did they also answer inconsistently?" This information could be used to find classes of users that had more than one inconsistent response- indicating that they might be an entirely different market segment. These users represent a segment who's needs are far from being met, and may the first to justify the creation of a new product.

Correcting Question Dialog that is Ambiguous

Investigating these confusion inconsistencies as well as elimination inconsistencies are useful to the marketer in finding questions that the user does not understand. Questions where users have the opportunity to be inconsistent can be closely scrutinized to determine if the questions are clear enough for the user to make an accurate response. It can be used to generate better questionnaires and reorganized how the recommendation set is positioned.

6.2.3 Dynamically ordered questions

The remaining questions can be analyzed to play "what if" scenarios. The expert can explore which questions have responses that show the most promise in propelling the user to a final recommendation by using some absolute measure of how close the expert is to a conclusion. In the Bayesian expert, the variance in a priori probability for each recommendation can be used.

To decide the order that the question panels will be shown in the questionnaire, a "question expert" is used to decide what will be shown next. This currently implemented as the question-Panel asking the question-Expert to return the next question-panel. This Trucktown version returns the question panels in the same order every time. In the next version of the question expert, the order of the question panels will be determined by what the next "best" question should be asked.

Because the user has the option to leave the questionnaire at any time, the expert will want to ask the most important questions as soon as possible. In the static ordered expert questionnaire, the order was chosen by a marketer to best mimic the typical order questions that a salesman might go through.

In a dynamically ordered expert questionnaire, the best next question to show will most likely:

1. Narrow the recommendation set.
2. Have few inconsistencies.
3. Invoke trust with the user.

Narrowing the recommendation set: A narrow recommendation set is one that has a high variance in probabilities of recommendations. Total variance can measured as $\mathbf{V = \Sigma\ |(P(K)-P(Kave))|/n}$. The following is a simple algorithm to find questions that have a high probability of narrowing the recommendation set:

1. Find the current variance in the recommendation set.
2. For each question that has not been responded to

3. Find the **change** in variance in the recommendation set for each possible response.
4. Weight these variance changes by the probability of how each question may be answered to find the Expected change of variance **if** the question is asked.

5. Rank the variance changes and/or take the question that will most likely **increase** the variance.

Minimizing Inconsistencies: It may be adventitious to avoid elimination inconsistencies in order to help steer the user into recommendations that have a high probability of closely fitting what the user is looking for. This strategy of leading the user to existing recommendations is helpful to current users. Unfortunately, this feature also reduces the usefulness of the inconstancy expert as a marketing tool (this marketing expert needs user inconsistencies to be able to find product holes and ambiguous dialog). The following is a simple algorithm to find questions that have a high probability of avoiding possible inconsistencies:

1. For each question that has not been responded to
    2. Find the **change** in the inconsistency score for each possible response.
    3. Weight these score changes by the probability of how each question may be answered to find the Expected change of variance **if** the question is asked.
4. Rank the changes and/or take the question that will most likely **decrease** the score.

This and the variance algorithms are breadth first and only go one layer deep. It is the simplest of all search algorithms. Going deeper in the tree of possible future responses can improve their performance. For example, instead of just searching the next question, the algorithm can look at the next 5 possible questions it can ask. This quickly becomes very computationally expensive (amount of computation expands exponentially with how deep the search is). If this became an area of research for the project, the expert should be put in its own thread and run in the background. It is a heuristic algorithm and is only looking to continually improve the on the score it already has.

Increasing Trust

A good recommendation expert is more than just a program that gives accurate recommendations, it is software that facilitates the buying process and increases the probability of a final purchase. In fact, the trust created by the recommendation process, rather than the accuracy of the recommendation, could be more dominant in influencing the user toward a buying decision. This depends on the user and the type of recommendation being given. A "good" recommendation of beer to a drunkard is less important than a "good" recommendation of furniture to match a finicky person's household decor.

There is an intangible notion of making the user feel more comfortable with the expert system. This can be shown with the following: The salesperson's first question could be "How are you doing today"? This question does not tell the salesperson/expert anything, but it does increase how comfortable the user is with the salesperson/expert system and will affect the likelihood of a final sale. How likely the user will take the expert's advice depends upon how well the user feels the expert understands their needs. Note that this is not necessarily the same thing as is the expert actually giving good advice.

Aside from the actual recommendation itself, the recommendation expert is useful for:
1. Building of trust by making the user feel that they are being listened to.
2. Allowing for response information to be used in the sale of the truck.

Using Trust to Influence the Expert Questionnaire

This notion that there is a trust factor associated with each question can be exploited in the implementation of the expert advisor. All other things being equal, the expert should ask the more highly rated trust-based questions first because the user has the opportunity to leave the questionnaire at any time.

Four types of questions are as follows:
1. How is your day? (a comfort-building question)

2. What kind of car are you interested in? (appears relevant, implicitly saying "I understand your needs")
3. Are you fat and ugly? (an alienating question)
4. Do you take your lunch to work? (doesn't appear relevant, implicitly saying "I don't understand your needs")

All four of these questions could be very helpful in bringing the expert come to a better recommendation. But asking the first two types of questions will help build trust, whereas the last two will lower trust. Even if the last two questions were asked (yielding a better recommendation), lowering user's comfort and trust could result in a lower probability that the better recommendation will be taken.

To use trust in the ordering of the questionnaire or , a measure of trust must first be defined. Possible ways to score the trust evoked by each question.

1. Users will leave the expert on a question that did not evoke trust in them.
2. Focus groups could be used to score trust in each question of the questionnaire.
3. Questions could be correlated with being asked and the user making any purchase, or in our study, if the user signs up for a test drive (difficult to measure if the questions are not asked in a random order during the study).

Question screens, rather than the questions themselves, are the items that trust is associated with. There are many factors associated with the expert questions that are difficult to measure, such as the dialog of the questions to the look and feel of the background. Often, several questions on a screen make it impossible to distinguish which question on the screen is influencing the user's trust.

Focus groups with marketer judgment can be used to make an initial score of the questions in the expert questionnaire. As a second phase of implementation, the questionnaire can be made dynamic by developing a method that correlates user's actions (that indicate trust) with specific question screens that have been asked. Finding these correlation's can then be used to update the trust scores of each question. The implementation of using this trust data on the fly will be part of the expert questions routine that determines what questions to ask next.

Unlike the other two examples of influencing the ordering of the questions, there are no dynamic algorithms for finding the trust score (it is done off-line after viewing user behavior in its entirety).

Combining These Scores to Generate the Next Question

These three influencing factors (and possibly others, such as the profit margins of each recommendation) are then combined to find the "best" next question to ask. The simplest way to combine the scores is to weight each factor, add the factors together for each question and the highest scored question is shown next (Ax + By + Cx). Other more complex ways of combining scores could be implemented, but the incremental value would be offset by confusion on the part of the expert marketer.

A caveat regarding the question expert is that it will "lead" the user into recommendations that are easy to find. There may be many local areas of "good" recommendations that will not be easily reached if this is expert is used.

There are more criteria that can be used in ordering the questions (or leading the user). For example, you could have a completely unbiased underlying recommendation engine, but choose questions that have the highest probability of leading the user to a favorable recommendation (such as high margin products, et cetera). Combined with using a sales expert, this expert system could prove to be a powerful way to market products electronically. The expert could even ask questions that have a high probability of casting a competitive product in a poor light, enabling the sales expert to illustrate their sales problems.

6.2.4 Other Expert Modals

Truck Correlation Expert

This advisor watches for actions that show whether or not a truck is a good recommendation. Because trucks are correlated with other trucks in utility and recommendation, the correlation expert can find other trucks to recommend based on this information. The analogy of this is when the salesman shows the user a car and asks, "What do you think of this truck?". The functionality of this expert could be implicitly incorporated into the current expert by adding these types of questions to the Bayesian expert.

Important Question Expert

Questions that the user feels are more important can be weighted more heavily in the recommendation expert. After all the questions are posed, the user is asked which questions they felt were particularly relevant to their needs. Another UI to receive this information is to have a "this is an important question to me" checkbox that appears on the screen of every question in the expert. And another way to implement the data collection of this expert is to present all of the questions to the user at once and then "watch" which questions are answered first.

Utility Point Expert

This is the most generic way of modeling an expert. The premise to this model is simple and easy to understand. For every response or action, each truck is rewarded or penalized some number of points. The truck with the most points is the recommended truck. This generic form of the expert is embodied in all the experts. For example, in the elimination model, each truck began with zero points and were doc'ed a point every time it was eliminated. Anything with a negative score was considered to be an eliminated truck.

This model could be made to learn over time. The initial point allocation system is derived directly from the marketer. As the system matures, the rules could be made more refined by using the following procedure:
1. If the expert recommendation is found to be "good", the user responses will give that recommendation a slightly **higher** rating for the next user.
2. If the expert recommendation is found to be "bad", the user responses will give that recommendation a slightly **lower** rating for the next user.

The utility point model can be extended to do more complex manipulations of the point allocation. An extreme example is the Bayesian Model, consisting of a generic way of manipulating points based on historical data. For the Bayesian Model, the point scale ranges from zero to one, will all of the scores adding up to one.

These other models could be added to the Logic expert and could be turned on or off, depending on what is found to work the best in user benchmark studies.

6.2.5 Alternative Initial A Priori Experts

There are several ways to begin the initial a priori value of each truck before the Bayesian expert changes the values. Four ways to derive the initial probabilities:
1. Popularity of trucks
2. Frequency of recommendations previously made by the expert advisor.
3. Utility analysis incorporating GM market research.
4. Using a combination of the above.

Using popularity of trucks is a strong way to derive the probability of best recommendations to give to a user. For example, if a truck has half the market share, the initial a priori probability that the truck is the "best" truck for the user would be 50%.

A possible problem with using popularity is the assumption that people buy the truck that best fit their needs. There are factors that influence the popularity of trucks but have nothing to do with the intrinsic value of the automobile. These factors include advertising, branding, and other non-tangible factors that influence the purchase decision. In fact, if the expert uses popularity as the initial a priori probability, the recommendation

will be simply the trucks that the user would have chosen without the help of an advisor. Though, this speeds up the process of selecting a truck, it does not necessarily help the user find a truck that "better" suits their needs.

Using frequency of different recommendations previously made by the expert advisor will yield good results only when the advisor has performed well in the past. This is a problem inherent to the entire Bayesian system. However, if the historical recommendations of the advisor have been "good", then using this data as the basis of making a priori probabilities, the solution is the optimal solution. How "good" recommendations are recognized by the system determines how well this strategy will perform.

Finally, a practical solution to integrating popularity with market strategy is to simply allow the marketer to populate an a priori database that reflects what the marketing expert believes is an accurate representation of the a priori probability of each recommendation.

6.2.6 Benchmarking to Systematically Improve the Expert Recommendations

The current version of the expert was not rigorously tested with real experts or end users. The questions generally "pushed" the recommendations in the correct "direction" and were deemed acceptable for the purposes of the general market research. The recommendation quality could be measured and tested against simulated users. From these tests, database input information can be changed and the engine can be altered to produce better recommendations with the same bank of simulated users. These alterations can be done by hand or an "Improvement Expert" could be created to dynamically make better recommendations (The dynamic Bayesian alters its database-input information).

6.2.7 Further Integration of Recommendation Engine with Entire Sales Process

Help screens, friendly data collecting GUI, and the "Consultative Selling Expert" are all features that help create customer "buy in" to the expert recommendations. They generally don't help with improving recommendation quality, but trust in and preference for this sales process is important for the technology to be commercially viable.

In the current implementation of the expert, the "consultative selling" functionality is the only sales process feature (other than the recommendations themselves) that is tailored to the individual user. Tailoring the sales process to different market segments can further pursue the philosophy of "mass customization".

Having a dynamically determined sales strategy could be implemented based on user interactions and responses. The user can be segmented by type of sales experience the environment should present, regardless of the specific recommendation. In Trucktown, the GM segment data could be used as a basis to generate the type of showroom sales experience.

The following is an example of one implementation of this philosophy: User responses and actions can be continually analyzed to decide how sophisticated and discriminating the user is. The non-discriminating buyer could be given only a few recommendations, whereas the discriminating buyer will want to be shown more options and be allowed to discard the poor recommendations. The expert questionnaire may be used to determine how picky the user is regarding their purchase.

One measure of user discrimination is how consistent the user was in answering the questionnaire to find existing trucks. If the user gives inconsistent responses in the expert questionnaire, it indicates that the user may be easily confused with the Trucktown buying process. This information can be used to tailor the rest of the sales process to that

individual. A discriminating person is concerned with making a **great** decision, and will many recommendations. A non-discriminating person is concerned with making just a **good** decision. The confused, non-discriminating user can be helped by guiding them through a more linear environment and be given fewer choices in the showroom.

It may be found that the above reasoning is completely backwards and this recommendation presentation interface strategy is non-optimal for their corresponding users. The strategy of customizing the sales process may be highly dependent on non-obvious variables and may be best researched with an experimental approach.

6.3 General Extensions to this Research

Of the general directions that this expert research can be extended, a few main avenues are enumerated as follows:

1. Investigate how this needs-based engine would be integrated with a complex specification-configuration engine- An underlying specification-configurator is necessary if an artificially intelligent expert advisor is to be able to tap a full range of recommendations that are possible. For example, in configuring a computer, the amount of RAM in the recommended configured computer could be made based on what activities the person will be using their recommended computer for. With a specification-based engine, the configurator will simply ask the user how much RAM they would like (with possibly giving the user an non-needed education in dynamic memory).

2. Develop new expert algorithms, exploring such tools as neural networks or solving such problems as the independent event assumption- Marketing research could be dovetailed with exploring the effectiveness of different recommendation algorithms. For example, an interesting analysis would be to build three fundamentally different underlying logical approaches- segmentation w/ utility logic, Bayesian logic, and

neural network logic. Each subject would use each expert with different underlying logic, and rank the accuracy of each expert.

3. Develop the expert tools that allow the marketer to analyze data- correlating response data, recommendations, and user actions- This research would attack the problem with having to use the subjective salesman in analyzing sales methodology and approach in a commercial environment. A deployed Trucktown environment would have tens of thousands of users every day. This mode of interaction simulates exactly one sales methodology, not the deployed interpretation of an army of salesmen. With a statistically significant subset of concurrent users, the marketer could have the ability to control the functionality of the program, such as substituting one type of showroom for another. While being able to discount all unknown variables, the marketer would observe the resulting differences in user actions (such as session length or the number of test-drive sign-ups per user).

4. Extend the expert from a primarily static expert to a dynamic learning expert- The underlying expert algorithms could be dynamically updated to generate better recommendations as more people go through the expert. The Bayesian expert already does this and the entire logic expert could be generalized to take advantage of this ability.

5. Develop a generic expert engine with tools that allow a marketer to apply their own interface, dialog, and recommendations- This exercise would be useful in exploring the problems that would arise in generalizing an expert engine for use in horizontal markets.

6.4 Extending the Java Implementation

If Trucktown is to remain an Internet Program, migration from the current programming environment could take one of four courses to combat the technical weaknesses of the current program.

1. Continue to extend the mostly-client-side integrated architecture with Java 1.1- This course will yield more functionality in the short term because it would not require a re-orienting architecture phase. This coarse of development may be appropriate if the research orientation becomes more concentrated on generating higher quality recommendations.
2. Replace and extend the mostly-client-side integrated architecture with Java 1.0- This course is the quickest way to commercialize the program to be used on the Internet. It is ubiquitously supported on all major browsers and platforms. If Trucktown will not be commercialized, this is a poor option as it is not the quickest way to develop a controlled research Internet project.
3. Replace the GUI with a more mature graphical environment and integrate with the Java client-side architecture- This is a radical shift in the course of development that may begin to yield good results within the next ½ a year. It relies on 3rd party venders supplying Director-like plugins or Active-X components that may or may not work well in an Internet Environment. This may be a good course of action if usability is found to be a dominant trust que.
4. Replace the mostly-client-side architecture with a more distributed server-side push architecture- This architecture may integrate much more easily with an existing Internet site. It is a step back from the integrated client-side environment that we currently have and would make the Trucktown to have a look and feel more akin to a standard Internet site. It is a longer, but much less risky course of development that may also integrate with standard GUIs much more easily. This coarse of development is excellent if standard navigation is found to be a strong trust que.

The recommendation logic of the expert has been sufficiently abstracted from the interface to be able to remain intact no matter which technical course is chosen. Because of the client-server architecture, the expert logic could be implemented in whatever language the AI programmer finds to be the most applicable tool.

6.5 Extensions Conclusion

Pursuing all of these research extensions would require several years for a full time developer to complete. The four main research extensions (recommendation quality, marketing tool, integration, and engine generalization) can be independently pursued. These research extensions are not synergistic and development in one will not help with development in another.

I recommend pursuing one area of research with the second research topic continually in mind. This will allow a deep investigation of the most important topic and prevent the technical side of the project from collapsing due to many requirements.

7. Final Conclusion

This thesis described the implementation and future extensions of a sales recommendation generator. The "consultative selling" behavior integrated the recommendation engine by customizing the sales process, responding to user information. The Java Internet environment proved excellent for the logic implementation of the implementation but poor for the GUI implementation. The expert functionality and code is an excellent base to extend the research and could be commercialized for use on the Internet.

Appendix

A. Logic Expert Code Listings

1. **LogicExpert.java**- Top level of the expert and contains all expert engines.
2. **BayesianExpert.java**- Contains the Utility Expert, Bayesian Expert, and Consultative Selling Expert algorithms.
3. **TopLayerResponseDB.java**- Contains the De-correlation Expert and Correlation Database.

```
1 package Trucktown.Expert;
2 import Trucktown.*;
3
4 //
5 //
6 // LogicExpert
7 //
8 //
9 public class LogicExpert{
10
11         private ExpertPanel expertPanel;
12
13         private BayesianExpert bayesianExpert;
14         //this should only be accessed by inconsistancy_panels
15         public InconsistencyExpert inconsistencyExpert;
16         //joes segment expert
17         private SegmentExpert segmentExpert;
18
19         //to see watch the inner logic of the expert
20         private StatusFrame statusFrame;
21         public boolean STATUSFRAME = false;
22         public boolean DEBUG = false;
23
24
25         //constructor
26         public LogicExpert(){
27                 bayesianExpert = new BayesianExpert();
28                 segmentExpert = new SegmentExpert();
29                 inconsistencyExpert = new InconsistencyExpert();
30
31                 if (STATUSFRAME){
32                         //nothing is abstracted- stubbed and will have to change the status frame la
   ter
33
34                                 //statusFrame = new StatusFrame(getParent());
35                                 //statusFrame.setBounds(730,530, 600, 800);
36                                 //statusFrame.show();
37                 }
38         }
39
40         //methods
41
42         //used by what im thinking panel
43     public Truck[] get4RecommendedBayesianAndSegmentTrucks(ResponsesDB responsesDB, Truck[] trucks){
44
45         double threshold = .05;
46
47         Truck[] topTrucks;
48         //get the top bayesian trucks from the bayesian expert
49         //also populates the bayesian salesFeatures
50         topTrucks = bayesianExpert.getTopBayesianTrucks(responsesDB, trucks);
51
52         //see if the top truck has a high enough a priori probability
53         if (topTrucks[0].prioriProb < threshold){
54             //not high enough, return no trucks
55             topTrucks[0] = null;
56             topTrucks[1] = null;
57             topTrucks[2] = null;
58             topTrucks[3] = null;
59             return topTrucks;
60         }
61
62         //replace the second two trucks with segment trucks (making sure that it wasn't one of the f
   irst trucks)
63         //ranks them by what is left over in the priori prob bayesian analysis and put the top bayes
   ians into the top 4
64         Truck[] segmentTrucks = segmentExpert.getTopSegmentTrucks(responsesDB, trucks);
65         int segmentIndex;
66         //load first segment truck
67         for (segmentIndex = 0;;segmentIndex++){
68             if ( (segmentTrucks[segmentIndex] != topTrucks[0]) &&
69                 (segmentTrucks[segmentIndex] != topTrucks[1]) ){
70
71                 topTrucks[2] = segmentTrucks[segmentIndex];
72                 break;
73             }
74         }
75         //load second truck
76         for (;;segmentIndex++){
77             if ( (segmentTrucks[segmentIndex] != topTrucks[0]) &&
78                 (segmentTrucks[segmentIndex] != topTrucks[1]) &&
79                 (segmentTrucks[segmentIndex] != topTrucks[2]) ){
80
81                 topTrucks[3] = segmentTrucks[segmentIndex];
82                 break;
83             }
84         }
85
86         //pop into these trucks a flag that indicates why it was chosen!!(assume that the features a
   re already there)
87         topTrucks[0].recommendationReason = Truck.BAYESIAN_REASON;
88         topTrucks[1].recommendationReason = Truck.BAYESIAN_REASON;
```

```
89
90          topTrucks[2].recommendationReason = Truck.SEGMENT_REASON;
91          topTrucks[3].recommendationReason = Truck.SEGMENT_REASON;
92
93          //put the sales features in right here!!!
94          topTrucks = bayesianExpert.makeSalesFeaturesProblems(topTrucks, responsesDB, trucks);
95
96          return topTrucks;
97      }
98
99           //hopefully, nothing was using this beast just updates all of the logic (well, this is a pr
    etty big deal
100         //check to see if i broke everything when i did this. (it may already be taken care of by t
    he get top trucks routine
101         /*
102         public Truck[] makeBestGuess(Truck[] trucks, ResponsesDB responsesDB){
103
104                 //defensive programming
105                 //check to see if the responsesDB and the respsonsesMatrixes in all the
106                 //truck objects match up (will keep AT LEAST the Bayesian Expert from breaking)
107                 //if there is a problem, it will print out and die
108                 if (DEBUG) {
109                     validateResponseData(trucks, responsesDB);
110                     //validate with a full set of dummie response data
111                     validateResponseData(trucks, responsesDB.getDummyResponsesDB());
112                 }
113                 //update the bayesian logic
114                 trucks = makeBestBayesianGuess(trucks);
115                 //update the segment logic.
116                 this.segmentArray = makeBestSegmentGuess(responsesDB);
117             //update segment truck data (in the sessionModel)
118             expertPanel.parentFrame.data.setSegmentTrucks(getSegmentTrucks(trucks));
119
120             //update the status frame from logic expert
121                 if (STATUSFRAME){
122                 //statusFrame.updateStatus(responsesDB, trucks, this.segmentArray);
123             }
124
125
126                 //ask the consistancy expert if we can show the next question
127                 //!!!!have to understand when this little beast will be used (after every question o
    r at the end?
128                 //!!!!should this be handled completely by the expert logic or should i possibly put
     it into the show
129                 //!!!!next question routine instead?
130                 //if (expertPanel.logicExpert.isConsistent()){
131                     //was consistant
132                 //}
133
134                 return trucks;
135     }
136
137         //make sure that all of these references are correct!!!!!!!!
138         */
139
140         public boolean isConsistent(ResponsesDB responsesDB){
141                 return inconsistencyExpert.isConsistent(responsesDB);
142         }
143
144         //overloaded method to abstract the responseDB for use with
145         public Truck[] makeBestBayesianGuess(Truck[] trucks, ResponsesDB rDB){
146                 return bayesianExpert.makeBestBayesianGuess(trucks, rDB);
147         }
148
149         public Truck[] getSegmentTrucks(ResponsesDB responsesDB,Truck[] trucks){
150             return segmentExpert.getSegmentTrucks(responsesDB, trucks);
151         }
152
153         public int[] makeBestSegmentGuess(ResponsesDB responses) {
154         return segmentExpert.makeBestSegmentGuess(responses);
155     }
156
157     //defensive programming
158     //check to see if the responsesDB and the respsonsesMatrixes in all the
159     //truck objects match up (will keep AT LEAST the Bayesian Expert from breaking)
160     //if there is a problem, it will print out a problem and die (will print out only the first prob
    lem it
161     //encounters
162         public void validateResponseData(Truck[] trucks, ResponsesDB responsesDB){
163         try{
164             //the data in the responsesDB is -1 or greater than 0
165             responsesDB.validateResponseArray();
166         }
167                 catch(NullPointerException e){
168                     error("problem with the reading the responsesDB");
169                 }
170
171
172             //iterate through the trucks
173             //check to see if the responseMatrix in the truck contains positive data in the correct
    places
174             //the responseDB has to be checked against each truck to see if it's response array will
```

```
    pull actual data
175         int i = 0;
176         try{
177             for (i = 0; i < trucks.length; i++){
178                 trucks[i].validateResponseData(responsesDB);
179             }
180         }
181                 catch(NullPointerException e){
182                     error("problem with the reading the trucks array at "+i);
183                 }
184         }
185
186         private void error(String message){
187                     System.out.println("logic expert error: " + message);
188                 System.exit(0);
189         }
190
191 }
192
```

```
1  package Trucktown.Expert;
2
3  import Trucktown.*;
4
5  //
6  //
7  // BayesianExpert
8  //assuming that e is 2.72
9  //
10 public class BayesianExpert {
11
12         private ResponsesDB rDB;
13
14         protected Truck[] trucks;
15
16     private static final boolean DEBUG = false;
17
18
19         //methods
20
21     //gets the top 4 trucks by probability (with reasons packed inside of them)
22     public Truck[] getTopBayesianTrucks(ResponsesDB rDB, Truck[] trucks){
23
24         //make sure that the bayesian update was made
25         trucks = makeBestBayesianGuess(trucks, rDB);
26
27         Truck[] topTrucks = new Truck[4];
28
29         //get the top truck first and put it in topTrucks
30         topTrucks[0] = trucks[0];
31         for(int trucksIndex = 1; trucksIndex<trucks.length; trucksIndex++){
32             if (trucks[trucksIndex].prioriProb > topTrucks[0].prioriProb){
33                 topTrucks[0] = trucks[trucksIndex];
34             }
35         }
36         //get the second top truck
37         double highestPrioriProbSoFar = 0;
38         //find the top truck that isn't already in topTrucks
39         for(int trucksIndex = 0; trucksIndex<trucks.length; trucksIndex++){
40             if (trucks[trucksIndex] == topTrucks[0]){
41                 //skip this truck if it is already in the DB
42             }
43             else if (trucks[trucksIndex].prioriProb < highestPrioriProbSoFar){
44                 //skip this truck if it is not the highest truck so far
45             }
46             else{
47                 //if here, cache this as the top truck
48                 highestPrioriProbSoFar = trucks[trucksIndex].prioriProb;
49                 topTrucks[1] = trucks[trucksIndex];
50             }
51         }
52
53         //get the third top truck
54         highestPrioriProbSoFar = 0;
55         for(int trucksIndex = 0; trucksIndex<trucks.length; trucksIndex++){
56             if (trucks[trucksIndex] == topTrucks[0]){
57                //skip this truck if it is already in the DB
58             }
59             else if (trucks[trucksIndex] == topTrucks[1]){}
60             else if (trucks[trucksIndex].prioriProb < highestPrioriProbSoFar){
61                 //skip this truck if it is not the highest truck so far
62             }
63             else{
64                 //if here, cache this as the top truck
65                 highestPrioriProbSoFar = trucks[trucksIndex].prioriProb;
66                 topTrucks[2] = trucks[trucksIndex];
67             }
68         }
69         //get the fourth top truck
70         highestPrioriProbSoFar = 0;
71         for(int trucksIndex = 0; trucksIndex<trucks.length; trucksIndex++){
72             //skip this truck if it is already in the DB
73             if (trucks[trucksIndex] == topTrucks[0]){}
74             else if (trucks[trucksIndex] == topTrucks[1]){}
75             else if (trucks[trucksIndex] == topTrucks[2]){}
76             //skip this truck if it is not the highest truck so far
77             else if (trucks[trucksIndex].prioriProb < highestPrioriProbSoFar){}
78             else {
79                 //if here, cache this as the top truck
80                 highestPrioriProbSoFar = trucks[trucksIndex].prioriProb;
81                 topTrucks[3] = trucks[trucksIndex];
82             }
83         }
84
85
86         //pack the reasons and problems inside of them
87         int truckIndex;
88         for (int i = 0;i<4;i++){
89             //get the index of the truck being investigated in the truck array
90             for (truckIndex = 0; truckIndex < trucks.length; truckIndex++){
91                 if (trucks[truckIndex] == topTrucks[i]){
92                     //found the index to pass on
```

```
93                      //get the reasons for that truck
94                      topTrucks[i].setBayesianSalesFeatures( this.makeBayesianSalesFeatures(rDB, truck
    s, truckIndex) );
95                      topTrucks[i].setBayesianSalesProblems( this.makeBayesianSalesProblemsAfterFeatur
    es(rDB, trucks, truckIndex) );
96                      }
97                  }
98          }
99
100         return topTrucks;
101     }
102
103         //returns the trucks with the updates
104         protected Truck[] makeBestBayesianGuess(Truck[] trucks, ResponsesDB rDB){
105                 this.trucks = trucks;
106                 this.rDB = rDB;
107
108         //utility analisis
109                 // Normalize answers
110                 normalizeChipResponses();
111                 // Populate prior probs
112                 populatePrioriProbs();
113                 //end utility analisis
114
115
116                 //xing
117                 //make the transfer from responsesFromPosedQuestions database to
118                 //responsesFromInnerModel database happen here
119                 //have it call a method in the responsesDB that does the data update!!!
120                 //that method will contain ALL of the ad hoc reasoning of filtering the responses
121                 //to the inner response DB
122
123
124
125                 //Make best guess iterating through all of the questions to update the priors

126                 makeBestGuess();
127
128                 //print();
129
130                 //the trucks now have the best prioris
131                 return this.trucks;
132         }
133
134         private void normalizeChipResponses(){
135                 //for each truck
136                 double totalNumberOfResponses = 0;
137                 for (int i = 0;i < trucks.length; i++){
138                         //check to see that the truck is not yet normalized
139                         if (!trucks[i].isNormalized){
140                                 //create the size of the normalized question array
141                                 trucks[i].normalizedResponseArrays = new double[trucks[i].numberOfRe
    sponseArrays.length][];
142
143                                 //for each q in that truck
144                                 for (int j = 1;j < trucks[i].normalizedResponseArrays.length;j++){
145                                         //allocate the space for the responses within this question
    array
146                                         trucks[i].normalizedResponseArrays[j] =
147                                                 new double[trucks[i].numberOfResponseArrays[j].lengt
    h];
148                                         //add up the # of responses over the question inside of the
    truck
149                                         totalNumberOfResponses = 0;
150                                         for (int k = 1;k < trucks[i].numberOfResponseArrays[j].lengt
    h; k++){
151                                                 totalNumberOfResponses =
152                                                         totalNumberOfResponses + trucks[i].numberOfR
    esponseArrays[j][k];
153                                         }
154                                         //for each answer
155                                         for (int k = 1;k < trucks[i].numberOfResponseArrays[j].lengt
    h; k++){
156                                                 //calculate normalized resposponse
157                                                 if (totalNumberOfResponses != 0){
158                                                         trucks[i].normalizedResponseArrays[j][k] =
159                                                                 (trucks[i].numberOfResponseArrays[j]
    [k]
160                                                                 / totalNumberOfResponses);
161                                                 }
162                                                 else{
163                                                         //would have been death divide by zero
164                                                 }
165                                         }
166                                 }//question loop
167                         }
168                         trucks[i].isNormalized = true;
169                 }//truck loop
170         }
171
172         private void populatePrioriProbs(){
173                 //calculate the utilities
```

```
174                 calculateUtilities();
175
176                 //calculate the choice  values from the utilities
177                 calculateChoiceValues();
178
179                 //calculate the priori's from the choice values
180                 calculatePrioris();
181         }
182
183         private void calculateUtilities(){
184                 //assume that the chips have been filled in
185
186                 //normalize the chip info in the response DB
187                 double totalChips = rDB.response_MRSP
188                                 + rDB.response_FuelEconomy
189                                 + rDB.response_Dependability
190                                 + rDB.response_Safety
191                                 + rDB.response_HorsePower;
192                 //use total chips to normalize
193                 rDB.normalizedResponse_MRSP = rDB.response_MRSP / totalChips;
194                 rDB.normalizedResponse_FuelEconomy = rDB.response_FuelEconomy / totalChips;
195                 rDB.normalizedResponse_Dependability = rDB.response_Dependability / totalChips;
196                 rDB.normalizedResponse_Safety = rDB.response_Safety / totalChips;
197                 rDB.normalizedResponse_HorsePower = rDB.response_HorsePower / totalChips;
198
199                 //simple addition of the utilities weighted with the chip allocations
200                 for (int i = 0;i < trucks.length; i++){
201                         trucks[i].utility = trucks[i].MRSP*rDB.normalizedResponse_MRSP +
202                                                         trucks[i].FuelEconomy*rDB.normalizedResponse
    _FuelEconomy +
203                                                         trucks[i].Dependability*rDB.normalizedRespon
    se_Dependability +
204                                                         trucks[i].Safety*rDB.normalizedResponse_Safe
    ty +
205                                                         trucks[i].HorsePower*rDB.normalizedResponse_
    HorsePower;
206                 }
207         }
208
209         private void calculateChoiceValues(){
210                 //1/(1 +e^utility)
211                 for (int i = 0;i < trucks.length; i++){
212                         trucks[i].choiceValue = 1/(1+ Math.exp(trucks[i].utility));
213                 }
214         }
215
216         private void calculatePrioris(){
217                 //add up all the choice values
218                 double totalChoiceValue = 0;
219                 for (int i = 0;i < trucks.length; i++){
220                         totalChoiceValue = totalChoiceValue + trucks[i].choiceValue;
221                 }
222                 //choice/totChoices
223                 for (int i = 0;i < trucks.length; i++){
224                         trucks[i].prioriProb = trucks[i].choiceValue/totalChoiceValue;
225                 }
226         }
227
228         private void makeBestGuess(){
229                 //for each question that was answered
230                 for (int questionIndex = 1;questionIndex < rDB.responseArray.length;questionIndex++)
    {
231                         //check that the question has been answered (-1 indicates that is wasn't fil
    led)
232                         if (-1 != rDB.responseArray[questionIndex].response){
233                                 //was not answered
234                                 //Update the priors using the question
235                                 updatePriorisOnce(questionIndex);
236                         }
237                 }//question loop
238
239
240         }
241
242         private void updatePriorisOnce(int questionIndex){
243                 //calculate the joint probability
244                 double jointProb = getJointProb(questionIndex);
245                 //make sure that jointProb is non-zero
246                 if (jointProb == 0){
247                        //make it just really small so the world doesn't blow up
248                        jointProb = .001;
249                 }
250                 //Calculate the new priors for each truck (pprob*normalizedAnswer/jointProb)
251                 for (int truckIndex = 0; truckIndex < trucks.length; truckIndex++){
252                         trucks[truckIndex].prioriProb = trucks[truckIndex].prioriProb
253                                 *trucks[truckIndex].normalizedResponseArrays[questionIndex][rDB.resp
    onseArray[questionIndex].response]
254                                 /jointProb;
255                 }
256         }
257
258         //suspect piece of code, tightly integrated and hard to test
```

```
259         //can return zero !!!
260         private double getJointProb(int questionIndex){
261                 double jointProb = 0;
262                 //(sum((normRArray)(priorprob)))
263                 for (int truckIndex = 0; truckIndex < trucks.length; truckIndex++){
264                         //
265             try{jointProb = jointProb +
266                                 (trucks[truckIndex].normalizedResponseArrays[questionIndex][rDB.resp
    onseArray[questionIndex].response]
267                                         * trucks[truckIndex].prioriProb);}
268                         catch(ArrayIndexOutOfBoundsException e)
269                         {
270                         }
271
272                 }
273
274                 return jointProb;
275         }
276
277
278     /*response expert figures out which update responses were the most influencial (ranks them.)
279     defined in the context of the rest of the responses ( doesn't include the utility sliders )
280     */
281
282     //note that this all depends and assumes that the a prioris are not changed at all
283     public Truck[] makeSalesFeaturesProblems(Truck[] recommendedTrucks, ResponsesDB responsesDB, Tru
    ck[] trucks){
284         //go through all of the recommended trucks
285         int truckIndex;
286         for(int recommendedIndex = 0 ; recommendedIndex < recommendedTrucks.length ; recommendedInde
    x++){
287             //get the index of the recommended truck in the trucks array
288             for (truckIndex = 0; truckIndex < trucks.length; truckIndex++){
289                 if (trucks[truckIndex] == recommendedTrucks[recommendedIndex]){
290                     //found the index to pass on
291                     //get the reasons for that truck
292                     recommendedTrucks[recommendedIndex].setBayesianSalesFeatures( this.makeBayesianS
    alesFeatures(responsesDB, trucks, truckIndex) );
293                     recommendedTrucks[recommendedIndex].setBayesianSalesProblems( this.makeBayesianS
    alesProblemsAfterFeatures(responsesDB, trucks, truckIndex) );
294                 }
295             }
296
297         }
298         return recommendedTrucks;
299     }
300
301     public String[] makeBayesianSalesFeatures(ResponsesDB rDB, Truck[] trucks, int truckIndex){
302
303         rDB = makeDeltasWithoutContext(rDB, trucks, truckIndex);
304         //rank the inner responses in order of positive influencers (using the delta)
305         rDB = rankResponsesBySalesDelta(rDB);
306         //get the sales features from the response objects in order of influence
307         String[] salesFeatures = getOrderedSalesFeatures(rDB);
308
309         //and return them
310         return salesFeatures;
311     }
312
313     public String[] makeBayesianSalesProblemsAfterFeatures(ResponsesDB rDB, Truck[] trucks, int truc
    kIndex){
314
315         //!!!!all this is commented out to be able to
316         //rDB = makeDeltasWithoutContext(rDB, trucks, truckIndex);
317         //rank the inner responses in order of positive influencers (using the delta)
318         //rDB = rankResponsesByDelta(rDB);
319         //get the sales problems from the response objects in order of influence
320         String[] salesProblems = getOrderedSalesProblems(rDB);
321
322
323         //and return them
324         return salesProblems;
325     }
326
327     public ResponsesDB makeDeltasWithContext(ResponsesDB rDB, Truck[] trucks, int truckIndex){
328
329         Truck recommendedTruck = trucks[truckIndex];
330
331         //run the responseDB and trucks through the bayesian expert to get the probability of the tr
    uck
332         trucks = makeBestBayesianGuess(trucks, rDB);
333
334         double baseProbability = recommendedTruck.prioriProb;
335         //go thru each question answered and score how influencial each question was by seeing the
336         //probability value when that response is missing
337
338         //!!!be carefull here! to make it go faster, I'm going to reuse the responseDb (picking out
    and
339         //saving the response being analized and then putting it back) (though i don't think that it
    's nes)
340         int cachedResponse;
341         //going through the responses
```

```
342         for (int responseIndex = 1; responseIndex < rDB.responseArray.length ;responseIndex++){
343             if (rDB.responseArray[responseIndex].response != -1){
344                 //this question was answered
345                 //remove and cache the response
346                 cachedResponse = rDB.responseArray[responseIndex].response;
347                 rDB.responseArray[responseIndex].response = -1;
348                 //run it through!
349                 trucks = makeBestBayesianGuess(trucks, rDB);
350                 //now store delta in the response
351                 //(prioriProb -baseProbability)\
352                 if (DEBUG){
353                     System.out.println("baseProb:"+baseProbability);
354                     System.out.println("priorprob without question:" + recommendedTruck.prioriProb);
355                 }
356                 rDB.responseArray[responseIndex].delta = baseProbability - recommendedTruck.prioriPr
    ob;
357                 //and put the response back
358                 rDB.responseArray[responseIndex].response= cachedResponse;
359             }
360             else{
361                 //response was -1 so make delta Response.NULL
362                 rDB.responseArray[responseIndex].delta = Response.NULL;
363             }
364         }
365 
366         return rDB;
367     }
368 
369     public ResponsesDB makeDeltasWithoutContext(ResponsesDB rDB, Truck[] trucks, int truckIndex){


370         Truck recommendedTruck = trucks[truckIndex];
371         //!!!!!!!!this breaks abstraction and will break if the inner logic changes!!!!!!!
372         //store the bayesian and utility responses in our temporary cashe(may want just stomp it all
     and then
373         //just translate the questions back down to the inner model instead
374         // to have better abstraction)!!!!
375 
376         //want to deal with responsesDB that has a clean slate   (!!!!!!!!clean the truck if needed
377         //clean out the deltas, rankings and sales features
378         rDB.cleanSalesInfo();
379 
380         int[] cachedResponses = new int[rDB.responseArray.length];
381         for (int i = 1; i < cachedResponses.length ;i++){
382             cachedResponses[i] = rDB.responseArray[i].response;
383         }
384 
385         double cachedResponse_MRSP = rDB.response_MRSP;
386         double cachedResponse_FuelEconomy = rDB.response_FuelEconomy;
387         double cachedResponse_Dependability =  rDB.response_Dependability;
388         double cachedResponse_Safety = rDB.response_Safety;
389         double cachedResponse_HorsePower = rDB.response_HorsePower;
390 
391         //create a baseline prioriProb for this truck. (xing!!!!the utility slider values were set t
    o 20-20-20-20-20
392         //to remove it's context!!!)
393         for (int i = 1; i < cachedResponses.length ;i++){
394             rDB.responseArray[i].response = -1;
395         }
396         rDB.response_MRSP = 20;
397         rDB.response_FuelEconomy = 20;
398         rDB.response_Dependability = 20;
399         rDB.response_Safety = 20;
400         rDB.response_HorsePower = 20;
401 
402 
403         //xing!!!!!!!!! this method is where you are going to deal with how the utility sliders affe
    cted the outcome
404 
405         //run the responseDB and trucks through the bayesian expert to get the base probability of t
    he truck
406         trucks = makeBestBayesianGuess(trucks, rDB);
407 
408         double baseProbability = recommendedTruck.prioriProb;
409         //go thru each question answered and score how influencial each question was by seeing the
410         //probability value when that response is the only one that is answered
411 
412         //!!!be carefull here! to make it go faster, I'm going to reuse the responseDb (putting in a
    nd
413         //then taking back out the response being analized
414         //going through the responses
415         //make the delta and the salesDelta
416         double delta = 0;
417         double salesDelta = 0;
418         Response respObj = null;
419         for (int responseIndex = 1; responseIndex < rDB.responseArray.length ;responseIndex++){
420             if (cachedResponses[responseIndex] != -1){
421                 //this question was answered
422                 respObj = rDB.responseArray[responseIndex];
423                 //add the response back
424                 respObj.response = cachedResponses[responseIndex];
425                 //run it through!
```

```
426                 trucks = makeBestBayesianGuess(trucks, rDB);
427                 //(prioriProb -baseProbability)
428                 delta = recommendedTruck.prioriProb - baseProbability;
429                 //added 4.10.98 by kjgl
430                 //make the salesDelta = delta* (response's sales weight)
431                 if (delta > 0){
432                     salesDelta = delta*(respObj.salesFeaturesWeight[respObj.response]);
433                 }
434                 else{
435                     salesDelta = delta*(respObj.salesProblemsWeight[respObj.response]);
436                 }
437                 //now store delta in the response
438                 rDB.responseArray[responseIndex].delta = delta;
439                 rDB.responseArray[responseIndex].salesDelta = salesDelta;
440
441
442                 //and remove the response again
443                 rDB.responseArray[responseIndex].response = -1;
444             }
445             else{
446                 //response was -1 so make delta Response.NULL
447                 rDB.responseArray[responseIndex].delta = Response.NULL;
448             }
449         }
450
451         //put everything back!
452         for (int i = 1; i < cachedResponses.length ;i++){
453             rDB.responseArray[i].response = cachedResponses[i];
454         }
455
456         rDB.response_MRSP = cachedResponse_MRSP;
457         rDB.response_FuelEconomy = cachedResponse_FuelEconomy;
458         rDB.response_Dependability = cachedResponse_Dependability;
459         rDB.response_Safety = cachedResponse_Safety;
460         rDB.response_HorsePower = cachedResponse_HorsePower;
461
462         //run the responseDB and trucks through the bayesian expert to get the a priori probs back i
    n order!!
463         trucks = makeBestBayesianGuess(trucks, rDB);
464
465         return rDB;
466     }
467
468     //only used by the status frame to look at the logic
469     /*ax out to make sure that the features are ranked by sales points
470
471     public ResponsesDB rankResponsesByDelta(ResponsesDB rDB){
472         //assume that the delta's are there and match up with what questions are answered!!
473
474         int currentRank;
475         //for each response
476         for (int responseIndex = 1; responseIndex < rDB.responseArray.length ;responseIndex++){
477             //go through the responses one by one and see how many responses had a higher value
478             //and make that plus one as their rank
479             if (rDB.responseArray[responseIndex].delta != Response.NULL){
480                 //rank this response
481                 currentRank = 1;
482                 for (int otherResponseIndex = 1; otherResponseIndex < rDB.responseArray.length ; oth
    erResponseIndex++){
483                     if ( rDB.responseArray[otherResponseIndex].delta != Response.NULL){
484                         //see if this other one has a higher rank
485                         if ( rDB.responseArray[otherResponseIndex].delta > rDB.responseArray[respons
    eIndex].delta){
486                             //this other response has a higher delta
487                             currentRank++;
488                         }
489                     }
490                 }
491             }
492             else{
493                 //delta was Response.NULL so the response goes rankless
494                 currentRank = -1;
495             }
496             //put the ranking into the response object
497             rDB.responseArray[responseIndex].rank = currentRank;
498         }//finished ranking
499
500         //go through and make sure that the ranks are unique
501         //by looking at the rank and doing a look forward through the list, incrementing all the lat
    er
502         //responses with the same rank (works because last bit looked like this {1,1,1,4,4,6) and NO
    T {3,3,3,5,5,6}
503         for (int responseIndex = 1; responseIndex < rDB.responseArray.length ;responseIndex++){
504             //go through rest of the responses and make the rest with the same rank one higher
505             if (rDB.responseArray[responseIndex].rank != -1){
506                 //see if there are any others with the same rank
507                 currentRank = rDB.responseArray[responseIndex].rank;
508                 for (int laterResponseIndex = responseIndex + 1; laterResponseIndex < rDB.responseAr
    ray.length ; laterResponseIndex++){
509                     if ( rDB.responseArray[laterResponseIndex].rank != -1){
510                         //see if this other one has the same rank
```

```
511                         if ( rDB.responseArray[laterResponseIndex].rank == rDB.responseArray[respons
    eIndex].rank){
512                             //up this later response has a lower rank!
513                             rDB.responseArray[laterResponseIndex].rank++;
514                         }
515                     }
516                 }//end looking at later ranks
517             }
518         }//end making ranking unique
519 
520         return rDB;
521     }
522     */
523 
524     //using the weighted delta's to make the sales points.
525     public ResponsesDB rankResponsesBySalesDelta(ResponsesDB rDB){
526         //assume that the delta's are there and match up with what questions are answered!!
527 
528         int currentRank;
529         //for each response
530         for (int responseIndex = 1; responseIndex < rDB.responseArray.length ;responseIndex++){
531             //go through the responses one by one and see how many responses had a higher value
532             //and make that plus one as their rank
533             if (rDB.responseArray[responseIndex].salesDelta != Response.NULL){
534                 //rank this response
535                 currentRank = 1;
536                 for (int otherResponseIndex = 1; otherResponseIndex < rDB.responseArray.length ; oth
    erResponseIndex++){
537                     if ( rDB.responseArray[otherResponseIndex].salesDelta != Response.NULL){
538                         //see if this other one has a higher rank
539                         if ( rDB.responseArray[otherResponseIndex].salesDelta > rDB.responseArray[re
    sponseIndex].salesDelta){
540                             //this other response has a higher salesDelta
541                             currentRank++;
542                         }
543                     }
544                 }
545             }
546             else{
547                 //salesDelta was Response.NULL so the response goes rankless
548                 currentRank = -1;
549             }
550             //put the ranking into the response object
551             rDB.responseArray[responseIndex].rank = currentRank;
552         }//finished ranking
553 
554         //go through and make sure that the ranks are unique
555         //by looking at the rank and doing a look forward through the list, incrementing all the lat
    er
556         //responses with the same rank (works because last bit looked like this {1,1,1,4,4,6) and NO
    T {3,3,3,5,5,6}
557         for (int responseIndex = 1; responseIndex < rDB.responseArray.length ;responseIndex++){
558             //go through rest of the responses and make the rest with the same rank one higher
559             if (rDB.responseArray[responseIndex].rank != -1){
560                 //see if there are any others with the same rank
561                 currentRank = rDB.responseArray[responseIndex].rank;
562                 for (int laterResponseIndex = responseIndex + 1; laterResponseIndex < rDB.responseAr
    ray.length ; laterResponseIndex++){
563                     if ( rDB.responseArray[laterResponseIndex].rank != -1){
564                         //see if this other one has the same rank
565                         if ( rDB.responseArray[laterResponseIndex].rank == rDB.responseArray[respons
    eIndex].rank){
566                             //up this later response has a lower rank!
567                             rDB.responseArray[laterResponseIndex].rank++;
568                         }
569                     }
570                 }//end looking at later ranks
571             }
572         }//end making ranking unique
573 
574         return rDB;
575     }
576 
577     private String[] getOrderedSalesFeatures(ResponsesDB rDB){
578 
579         //size the returning salesFeatures array
580         int arraySize = 1;
581         //for each response that is ranked and is possitive!
582         for (int responseIndex = 1; responseIndex < rDB.responseArray.length ;responseIndex++){
583             //go through the responses and increase size based on what is ranked!
584             if ( (rDB.responseArray[responseIndex].rank != -1) && (rDB.responseArray[responseIndex].
    salesDelta > 0) ){
585                 //count this response
586                 arraySize++;
587             }
588         }//end counting
589         String salesFeatures[] = new String[arraySize];
590 
591         //go through the ranked responses and fill in the correct places in the returned array (rank
     1 goes in index 0!!)
592         Response respObject;
593         for (int responseIndex = 1; responseIndex < rDB.responseArray.length ;responseIndex++){
```

```
            //go through the responses that have possitive salesDeltas
            respObject = rDB.responseArray[responseIndex];
            if ((respObject.rank != -1) && (respObject.salesDelta > 0)){
                //enter the sales feature into the correct place
                salesFeatures[respObject.rank - 1] = respObject.salesFeatures[respObject.response];
            }
        }//end filling in the sales features

        return salesFeatures;
    }

    private String[] getOrderedSalesProblems(ResponsesDB rDB){

        double problemStrength = -.002;

        //size the returning salesProblems array
        int arraySize = 0;//ends up being the number of problems
        int numberRanked = 0;//ends up being the number of questions answered (# of worst ranking)
        //for each response that is ranked and is negative!
        for (int responseIndex = 1; responseIndex < rDB.responseArray.length ;responseIndex++){
            //go through the responses and increase size based on what is ranked!
            if ( (rDB.responseArray[responseIndex].rank != -1) && (rDB.responseArray[responseIndex].
salesDelta < problemStrength ) ){
                //count this response
                arraySize++;
            }
            if (rDB.responseArray[responseIndex].rank != -1){
                numberRanked++;
            }
        }//end counting
        String salesProblems[] = new String[arraySize];

        //go through the ranked responses and fill in the correct places in the returned array (wors
t ranked goes in index 0!!)
        Response respObject;
        for (int responseIndex = 1; responseIndex < rDB.responseArray.length ;responseIndex++){
            //go through the responses to find the one with negative salesDeltas
            respObject = rDB.responseArray[responseIndex];
            if ((respObject.rank != -1) && (respObject.salesDelta < problemStrength)){
                //enter the sales feature into the correct place
                try{
                    salesProblems[numberRanked - respObject.rank] = respObject.salesProblems[respObj
ect.response];
                }
                catch (Exception e){
                    System.out.println("oh shit:"+e);
                }
            }
        }//end filling in the sales problems

        return salesProblems;
    }


        //test routine
        private void print(){

                System.out.println("MRSP: "+rDB.normalizedResponse_MRSP +"::"+rDB.response_MRSP);
                System.out.println("fuel economy: "+    rDB.normalizedResponse_FuelEconomy+"::"+rDB.
response_FuelEconomy);
                System.out.println("dependibility: "+rDB.normalizedResponse_Dependability+"::"+rDB.r
esponse_Dependability);
                System.out.println("safety: "+rDB.normalizedResponse_Safety+"::"+rDB.response_Safety
);
                System.out.println("horsePower: "+rDB.normalizedResponse_HorsePower+"::"+rDB.respons
e_HorsePower);
        }
}
```

```
1  package Trucktown.Expert;
2
3  public class TopLayerResponseDB {
4
5      public int[] directAnswers;
6
7      ResponsesDB innerResponsesDB;
8          public int[] responseArray;
9          public static int NUMBEROFTOPQUESTIONS = 48;
10         //deal with chip allocation answers in a speacial way(asuming that they
11     //are not going to be used in the bayesian update
12     //default them to be 20
13         public double response_MRSP = 20;
14         public double response_FuelEconomy = 20;
15         public double response_Dependability = 20;
16         public double response_Safety = 20;
17         public double response_HorsePower = 20;
18
19         //to be calculated once the chip responses are filled in.
20         public double normalizedResponse_MRSP;
21         public double normalizedResponse_FuelEconomy;
22         public double normalizedResponse_Dependability;
23         public double normalizedResponse_Safety;
24         public double normalizedResponse_HorsePower;
25
26         //where to put the segment array that the segment expert gives back
27         public int[] cachedSegArray = {1,2,3,4,5,6,7,8};
28
29     //values that give meaning to the response Database
30     public static final int NULL_TOP                    = 0;
31     public static final int FULLORCOMPACT_TOP           = 1;    // question panel 1 (3 vals)
32     public static final int PRICERANGE_TOP              = 2;    // question panel 5 (8 vals)
33     public static final int ROUGHROAD_TOP               = 3;    // question panel 3 (2 TOPs ea.)
34     public static final int DRIVEONICEORSNOW_TOP        = 4;    // question panel 3
35     public static final int TOWTRAILER_TOP              = 5;    // question panel 3
36     public static final int OFFROADDRIVING_TOP          = 6;    // question panel 3
37     public static final int FISHERORHUNTER_TOP          = 7;    // question panel 3
38     public static final int HOMESUPPLYHAULING_TOP       = 8;    // question panel 3
39     public static final int GENERALHAULING_TOP          = 9;    // question panel 3
40     public static final int CONSTRUCTION_TOP            = 10;   // question panel 4 (2 vals ea.)
41     public static final int TOWING_TOP                  = 11;   // question panel 4
42     public static final int SNOWPLOWING_TOP             = 12;   // question panel 4
43     public static final int HAULING_TOP                 = 13;   // question panel 4
44     public static final int NUMBERPEOPLE_TOP            = 14;   // question panel 2 (5 vals)
45     public static final int PEOPLEINFRONTSEAT_TOP       = 15;   // question panel 8 (3 vals)
46     public static final int TRUCKSIZE_TOP               = 16;   // question panel 9 (5 vals)
47     public static final int LIKESCHEVY_TOP              = 17;   // 17-24: question panel 6 (2 vals e
   a.)
48     public static final int LIKESDODGE_TOP              = 18;
49     public static final int LIKESFORD_TOP               = 19;
50     public static final int LIKESGMC_TOP                = 20;
51     public static final int LIKESISUZU_TOP              = 21;
52     public static final int LIKESMAZDA_TOP              = 22;
53     public static final int LIKESNISSAN_TOP             = 23;
54     public static final int LIKESTOYOTA_TOP             = 24;
55     public static final int BEDSIZE_TOP                 = 25;   // question panel 11 (3 vals)
56     public static final int TALLESTUSER_TOP             = 26;   // question panel 10 (3 vals)
57     public static final int FOURCYLGASENGINE_TOP        = 27;   // question panel 12 (2 vals ea.)
58     public static final int SIXCYLGASENGINE_TOP         = 28;   // question panel 12
59     public static final int EIGHTCYLGASENGINE_TOP       = 29;   // question panel 12
60     public static final int TENCYLGASENGINE_TOP         = 30;   // question panel 12
61     public static final int EIGHTCYLDIESEL_TOP          = 31;   // question panel 12
62     public static final int DONTKNOWENGINE_TOP          = 32;   // question panel 12
63     public static final int COMFORT_TOP                 = 33;   // question panel 13
64     public static final int BODYSTYLING_TOP             = 34;   // question panel 14
65     public static final int GREATSTERIO_TOP             = 35;   // 35-47: options panel
66         public static final int THRDFRTHDOOR_TOP            = 36;
67         public static final int LEATHERSEATS_TOP            = 37;
68         public static final int CRUISECONTROL_TOP           = 38;
69         public static final int POWERLOCK_TOP               = 39;
70         public static final int POWERWINDOW_TOP             = 40;
71         public static final int TILTWHEEL_TOP               = 41;
72         public static final int TINTWINDOW_TOP              = 42;
73         public static final int THEFTDETER_TOP              = 43;
74         public static final int SAFETYUNIMP_TOP             = 44;
75         public static final int BETTERTIRES_TOP             = 45;
76         public static final int ANTILOCKBRAKES_TOP          = 46;
77         public static final int FOURWHEELDR_TOP             = 47;
78         public static final int AUTOMATIC_TOP               = 48;
79
80
81     public static int[] TopToLowerTable;
82
83     static {
84         TopToLowerTable = new int[(NUMBEROFTOPQUESTIONS+1)];
85         TopToLowerTable[NULL_TOP] = ResponsesDB.NULL_VAL;
86         TopToLowerTable[FULLORCOMPACT_TOP] = ResponsesDB.FULLORCOMPACT_VAL;
87         TopToLowerTable[PRICERANGE_TOP] = ResponsesDB.PRICERANGE_VAL;
88         TopToLowerTable[ROUGHROAD_TOP] = ResponsesDB.OFFROAD_VAL;
89         TopToLowerTable[DRIVEONICEORSNOW_TOP] = ResponsesDB.OFFROAD_VAL;
90         TopToLowerTable[TOWTRAILER_TOP] = ResponsesDB.TOW_VAL;
91         TopToLowerTable[OFFROADDRIVING_TOP] = ResponsesDB.OFFROAD_VAL;
```

```
92          TopToLowerTable[FISHERORHUNTER_TOP] = ResponsesDB.OFFROAD_VAL;
93          TopToLowerTable[HOMESUPPLYHAULING_TOP] = ResponsesDB.HAULING_VAL;
94          TopToLowerTable[GENERALHAULING_TOP] = ResponsesDB.HAULING_VAL;
95          TopToLowerTable[CONSTRUCTION_TOP] = ResponsesDB.CONSTRUCTION_VAL;
96          TopToLowerTable[TOWING_TOP] = ResponsesDB.TOW_VAL;
97          TopToLowerTable[SNOWPLOWING_TOP] = ResponsesDB.SNOWPLOWING_VAL;
98          TopToLowerTable[HAULING_TOP] = ResponsesDB.HAULING_VAL;
99          TopToLowerTable[NUMBERPEOPLE_TOP] = ResponsesDB.NUMBERPEOPLE_VAL;
100         TopToLowerTable[PEOPLEINFRONTSEAT_TOP] = ResponsesDB.PEOPLEINFRONTSEAT_VAL;
101         TopToLowerTable[TRUCKSIZE_TOP] = ResponsesDB.TRUCKSIZE_VAL;
102         TopToLowerTable[LIKESCHEVY_TOP] = ResponsesDB.LIKESCHEVY_VAL;
103         TopToLowerTable[LIKESFORD_TOP] = ResponsesDB.LIKESFORD_VAL;
104         TopToLowerTable[LIKESDODGE_TOP] = ResponsesDB.LIKESDODGE_VAL;
105         TopToLowerTable[LIKESGMC_TOP] = ResponsesDB.LIKESGMC_VAL;
106         TopToLowerTable[LIKESISUZU_TOP] = ResponsesDB.LIKESISUZU_VAL;
107         TopToLowerTable[LIKESMAZDA_TOP] = ResponsesDB.LIKESMAZDA_VAL;
108         TopToLowerTable[LIKESNISSAN_TOP] = ResponsesDB.LIKESNISSAN_VAL;
109         TopToLowerTable[LIKESTOYOTA_TOP] = ResponsesDB.LIKESTOYOTA_VAL;
110         TopToLowerTable[BEDSIZE_TOP] = ResponsesDB.BEDSIZE_VAL;
111         TopToLowerTable[TALLESTUSER_TOP] = ResponsesDB.TALLESTUSER_VAL;
112         TopToLowerTable[FOURCYLGASENGINE_TOP] = ResponsesDB.FOURCYLGASENGINE_VAL;
113         TopToLowerTable[SIXCYLGASENGINE_TOP] = ResponsesDB.GREATSTERIO_VAL;
114         TopToLowerTable[EIGHTCYLGASENGINE_TOP]=ResponsesDB.EIGHTCYLGASENGINE_VAL;
115         TopToLowerTable[TENCYLGASENGINE_TOP]=ResponsesDB.TENCYLGASENGINE_VAL;
116         TopToLowerTable[EIGHTCYLDIESEL_TOP] = ResponsesDB.EIGHTCYLDIESEL_VAL;
117         TopToLowerTable[DONTKNOWENGINE_TOP]=ResponsesDB.NULL_VAL;
118         TopToLowerTable[COMFORT_TOP]=ResponsesDB.COMFORT_VAL;
119         TopToLowerTable[BODYSTYLING_TOP]=ResponsesDB.BODYSTYLING_VAL;
120         TopToLowerTable[GREATSTERIO_TOP]=ResponsesDB.GREATSTERIO_VAL;
121         TopToLowerTable[THRDFRTHDOOR_TOP] = ResponsesDB.THRDFRTHDOOR_VAL;
122         TopToLowerTable[LEATHERSEATS_TOP] = ResponsesDB.LEATHERSEATS_VAL;
123         TopToLowerTable[CRUISECONTROL_TOP] = ResponsesDB.CRUISECONTROL_VAL;
124         TopToLowerTable[POWERLOCK_TOP] = ResponsesDB.POWERLOCK_VAL;
125         TopToLowerTable[POWERWINDOW_TOP] = ResponsesDB.POWERWINDOW_VAL;
126         TopToLowerTable[TILTWHEEL_TOP] = ResponsesDB.TILTWHEEL_VAL;
127         TopToLowerTable[TINTWINDOW_TOP] = ResponsesDB.TINTWINDOW_VAL;
128         TopToLowerTable[THEFTDETER_TOP] = ResponsesDB.THEFTDETER_VAL;
129         TopToLowerTable[SAFETYUNIMP_TOP] = ResponsesDB.SAFETYUNIMP_VAL;
130         TopToLowerTable[BETTERTIRES_TOP] = ResponsesDB.BETTERTIRES_VAL;
131         TopToLowerTable[ANTILOCKBRAKES_TOP] = ResponsesDB.ANTILOCKBRAKES_VAL;
132         TopToLowerTable[FOURWHEELDR_TOP] = ResponsesDB.FOURWHEELDR_VAL;
133         TopToLowerTable[AUTOMATIC_TOP] = ResponsesDB.AUTOMATIC_VAL;
134     }
135
136     public TopLayerResponseDB(ResponsesDB InnerDB) {
137         innerResponsesDB = InnerDB;
138             directAnswers = new int[(NUMBEROFTOPQUESTIONS+1)];
139             for(int i = 0; i < directAnswers.length; i++) {
140                 directAnswers[i] = -1;
141             }
142     }
143
144     //set and get methods
145     public void setTLVal(int index, int answer) {
146         directAnswers[index] = answer;
147         if(index==TOWING_TOP || index==TOWTRAILER_TOP) {
148             updateTowing();
149         }
150         else if(index==GENERALHAULING_TOP || index ==HAULING_TOP
151             ||index == HOMESUPPLYHAULING_TOP) {
152             updateHauling();
153         }
154         else if(index==ROUGHROAD_TOP || index == DRIVEONICEORSNOW_TOP
155                 || index==OFFROADDRIVING_TOP || index==FISHERORHUNTER_TOP) {
156             updateOffRoad();
157         }
158         else if(index==DONTKNOWENGINE_TOP) {}
159         else innerResponsesDB.responseArray[TopToLowerTable[index]].response = answer;
160
161     }
162     public int getTLVal(int index) {
163         return directAnswers[index];
164     }
165
166     public void set_response_MRSP(int value){
167         response_MRSP = value;
168         innerResponsesDB.response_MRSP = value;
169     }
170
171     public void set_response_FuelEconomy(int value){
172         response_FuelEconomy = value;
173         innerResponsesDB.response_FuelEconomy = value;
174     }
175
176     public void set_response_Dependability(int value){
177         response_Dependability = value;
178         innerResponsesDB.response_Dependability = value;
179     }
180
181     public void set_response_Safety(int value){
182         response_Safety = value;
183         innerResponsesDB.response_Safety = value;
```

```
184     }
185
186     public void set_response_HorsePower(int value){
187         response_HorsePower = value;
188         innerResponsesDB.response_HorsePower = value;
189     }
190
191
192     //pushes all of the information in the top layer into the lower layer
193     public void updateInnerResponsesDB() {
194         for(int i = 1; i<directAnswers.length; i++) {
195             if (TopToLowerTable[i] != 0){
196                 //this answer can be filtered down to an existing response object
197                 innerResponsesDB.responseArray[TopToLowerTable[i]].response = directAnswers[i];

198             }
199         }
200         updateHauling();
201         updateTowing();
202         updateOffRoad();
203         updateSliders();
204     }
205
206     //lower level updates
207     public void updateSliders() {
208          innerResponsesDB.response_MRSP = response_MRSP;
209          innerResponsesDB.response_FuelEconomy = response_FuelEconomy;
210          innerResponsesDB.response_Dependability = response_Dependability;
211          innerResponsesDB.response_Safety = response_Safety;
212          innerResponsesDB.response_HorsePower = response_HorsePower;
213     }
214
215
216
217     public void updateHauling() {
218         if(directAnswers[HOMESUPPLYHAULING_TOP] ==2 || directAnswers[GENERALHAULING_TOP] ==2 || dire
    ctAnswers[HAULING_TOP]==2) {
219             innerResponsesDB.responseArray[ResponsesDB.HAULING_VAL].response=2;
220         }
221         else {
222             innerResponsesDB.responseArray[ResponsesDB.HAULING_VAL].response=1;
223         }
224     }
225
226     public void updateTowing() {
227         if(directAnswers[TOWING_TOP] ==2 || directAnswers[TOWTRAILER_TOP]==2) {
228             innerResponsesDB.responseArray[ResponsesDB.TOW_VAL].response=2;
229         }
230         else {
231             innerResponsesDB.responseArray[ResponsesDB.TOW_VAL].response=1;
232         }
233     }
234
235     public void updateOffRoad() {
236         if(directAnswers[ROUGHROAD_TOP]==2 || directAnswers[DRIVEONICEORSNOW_TOP] == 2
237            || directAnswers[OFFROADDRIVING_TOP]==2 || directAnswers[FISHERORHUNTER_TOP]==2)
238         {
239             innerResponsesDB.responseArray[ResponsesDB.OFFROAD_VAL].response=2;
240         }
241         else {
242             innerResponsesDB.responseArray[ResponsesDB.OFFROAD_VAL].response=1;
243         }
244     }
245
246         public void print(){
247             for (int i = 1; i<directAnswers.length; i++){
248                     System.out.println("answer_:" +i+": "+ directAnswers[i]);
249                 }
250         }
251
252 }
```

B. GUI Code Listings

1. **ExpertPanel.java**- Is the Expert Module, containing all expert GUI and logic functionality.
2. **MechanicExpertIntroPanel.java**- One of the three expert panels that the user encounters when entering an expert.
3. **QuestionsPanel.java**- controls the navigation and showing of the expert questions.
4. Question_1Panel.java- One of the sixteen question panels.
5. **QuestionInterface.java**- Allows the QuestionsPanel to handle the **Question_#Panel** in a standard and flexible manner.
6. **DoneWithQuestions.java**- Controls the exiting navigation.
7. **NeedMoreAnswersPanel.java**- Tells the user that they did not answer enough questions.
8. **ExpertExitPanel.java**- Tells the user "goodbye".
9. **ExpertReturningPanel.java**- Tells the user "hello again".
10. **BayesianSegmentRoom.java**- Retrieves and displays the recommended trucks in icon format.
11. **TruckPanel.java**- Displays a recommended truck and allows navigation to more information.
12. **WhatImThinkingPanel.java**- Controls the display of the current recommendations with recommendation explanations.
13. **WITPMainPanel.java**- Displays the current recommended trucks in icon format.
14. **WITPTruckPanel.java**- Displays a recommended truck and explanation.
15. **FeaturesPanel.java**- Used to display the sales features and problems of a recommended truck.
16. **StatusFrame.java**- Used for debugging.

```
1  package Trucktown.Expert;
2  import Trucktown.*;
3  import java.awt.*;
4
5  /* this is to be turned into 3 experts that essentially look identical
6      the differences are the intro and exit panels, background images and the
7      update to the responseDB of which expert the user chose.  the same logic updates are
8      used.
9      */
10
11 public class ExpertPanel extends Panel{
12
13         TrucktownFrame parentFrame;
14
15     //common expert objects.
16     //background photo
17     protected Image expertBackground;
18
19 //    protected Color buttonColor = new Color(16762880);
20     protected Color buttonColor = new Color(206,206,206);
21     protected Color helpButColor = Color.green;
22     protected Font smallFont = new Font("Dialog", Font.BOLD, 14);
23     protected Font bigFont = new Font("Dialog", Font.BOLD, 20);
24
25         //classes used by the expert
26         protected ResponsesDB responsesDB;
27
28         protected TopLayerResponseDB topLayerDB;//added by xing 3/12/98
29
30         protected LogicExpert logicExpert;
31         protected QuestionsPanel questionsPanel;
32         protected ExpertHistory expertHistory;
33         public SessionModel data;
34
35     //expertType is to differentiate the experts
36     //would have liked to have made different expert that extend expert BUT would have gotten
37     //into casting hell
38     protected int expertType;
39     public static final int NONE        = 0;
40     public static final int MECHANIC    = 1;
41     public static final int NEIGHBOR    = 2;
42     public static final int EDITOR      = 3;
43
44         //list of trucks to be evaluated
45         protected Truck[] trucks;
46
47         //deal with returning users
48         private boolean shownBefore = false;
49
50         //state of the questionnaire (has the person finished?)
51         private int state;
52
53         //these test booleans are to be used everywhere
54         public boolean DEBUG = false;
55
56         //put in to get the status frame working
57         StatusFrame statusFrame;
58
59     private String[] mechanicQuestionPanelArray= {"",
60                         "Trucktown.Expert.Question_1Panel",
61                         "Trucktown.Expert.Question_2Panel",
62                         "Trucktown.Expert.Question_3Panel",
63                         "Trucktown.Expert.Question_4Panel",
64                         "Trucktown.Expert.Question_5Panel",
65                         "Trucktown.Expert.Question_6Panel",
66                         "Trucktown.Expert.Question_7Panel",
67                         "Trucktown.Expert.Question_8Panel",
68                         "Trucktown.Expert.Question_9Panel",
69                         "Trucktown.Expert.Question_10Panel",
70                         "Trucktown.Expert.Question_11Panel",
71                         "Trucktown.Expert.Question_12Panel",
72                         "Trucktown.Expert.Question_16Panel",
73                         "Trucktown.Expert.Question_13Panel",
74                         "Trucktown.Expert.Question_14Panel",
75                         "Trucktown.Expert.Question_15Panel"};
76
77     private String[] neighborQuestionPanelArray= {"",
78                         "Trucktown.Expert.Question_1Panel",
79                         "Trucktown.Expert.Question_2Panel",
80                         "Trucktown.Expert.Question_3Panel",
81                         "Trucktown.Expert.Question_4Panel",
82                         "Trucktown.Expert.Question_5Panel",
83                         "Trucktown.Expert.Question_6Panel",
84                         "Trucktown.Expert.Question_7Panel",
85                         "Trucktown.Expert.Question_8Panel",
86                         "Trucktown.Expert.Question_9Panel",
87                         "Trucktown.Expert.Question_10Panel",
88                         "Trucktown.Expert.Question_11Panel",
89                         "Trucktown.Expert.Question_12Panel",
90                         "Trucktown.Expert.Question_16Panel",
91                         "Trucktown.Expert.Question_13Panel",
92                         "Trucktown.Expert.Question_14Panel",
```

```
                        "Trucktown.Expert.Question_15Panel"};

    private String[] editorQuestionPanelArray= {"",
                        "Trucktown.Expert.Question_1Panel",
                        "Trucktown.Expert.Question_2Panel",
                        "Trucktown.Expert.Question_3Panel",
                        "Trucktown.Expert.Question_4Panel",
                        "Trucktown.Expert.Question_5Panel",
                        "Trucktown.Expert.Question_6Panel",
                        "Trucktown.Expert.Question_7Panel",
                        "Trucktown.Expert.Question_8Panel",
                        "Trucktown.Expert.Question_9Panel",
                        "Trucktown.Expert.Question_10Panel",
                        "Trucktown.Expert.Question_16Panel",
                        "Trucktown.Expert.Question_11Panel",
                        "Trucktown.Expert.Question_12Panel",
                        "Trucktown.Expert.Question_13Panel",
                        "Trucktown.Expert.Question_14Panel",
                        "Trucktown.Expert.Question_15Panel"};

        //constructor
        public ExpertPanel(TrucktownFrame parentFrame, int expertType){
                this.parentFrame = parentFrame;
                this.data = this.parentFrame.data;
                this.expertType = expertType;
                //load the image that will be in every background
                if (expertType == this.MECHANIC){
            expertBackground = parentFrame.util.getImage( "Images/Expert/garage_expert.jpg" );
        }
        else if (expertType == this.NEIGHBOR){
            expertBackground = parentFrame.util.getImage( "Images/Expert/neighbor_expert.jpg" );
        }
        else if (expertType == this.EDITOR){
            expertBackground = parentFrame.util.getImage( "Images/Expert/editor_expert.jpg" );
        }
        }

        //show it and do it!!
        //requires truck objects as inputs
    //    gets truck array and responseDB from SessionModel and calls showExpert()
        public void show(){
                if (expertType == this.MECHANIC){
                this.trucks          = data.getAllTrucks();
            this.responsesDB    = data.responsesDB;
            this.topLayerDB = data.mechanic_topLayerDB; //added by xing 3/13/98

            //stomp the lower level response set by filtering the
            //the top layer responses down.
            this.topLayerDB.updateInnerResponsesDB();

            //deal with returning users
            if (shownBefore){
                showAgain();
            }
            else{
                ////create and show everything for the first time

                        //make the logic expert
                        logicExpert = new LogicExpert();

                        //make the questions panel (added! this may be a problem)
                        questionsPanel = new QuestionsPanel(this, mechanicQuestionPanelArray);
                        this.add(questionsPanel);

                        //make the history panel
                    expertHistory = data.mechanic_expertHistory;

                        //make the intro panel
                        MechanicExpertIntroPanel expertIntroPanel = new MechanicExpertIntroPanel(thi
s);
                        this.add(expertIntroPanel);

                        //show the intro panel
                        //really different for every expert
                        expertIntroPanel.showExpertIntro();
                        //show me!   Commented out by ART on 11-16
                        super.show();
                this.shownBefore = true;
                }
        }
        else if (expertType == this.NEIGHBOR){
                this.trucks          = data.getAllTrucks();
            this.responsesDB    = data.responsesDB;
            this.topLayerDB = data.neighbor_topLayerDB; //added by xing 3/13/98

            //stomp the lower level response set by filtering the
            //the top layer responses down.
            this.topLayerDB.updateInnerResponsesDB();

            //deal with returning users
            if (shownBefore){
                showAgain();
```

```
184             }
185             else{
186                 ////create and show everything for the first time
187 
188                         //make the logic expert
189                         logicExpert = new LogicExpert();
190 
191                 //make a status frame
192                                 statusFrame = new StatusFrame(this);
193                                 statusFrame.setBounds(730,530, 600, 800);
194                                 statusFrame.show();
195 
196 
197                         //make the questions panel (added! this may be a problem)
198                         questionsPanel = new QuestionsPanel(this, neighborQuestionPanelArray);
199                         this.add(questionsPanel);
200 
201                     expertHistory = data.neighbor_expertHistory;
202 
203                         //make the intro panel
204                         NeighborExpertIntroPanel expertIntroPanel = new NeighborExpertIntroPanel(thi
    s);
205                         this.add(expertIntroPanel);
206 
207                         //show the intro panel
208                         //really different for every expert
209                         expertIntroPanel.showExpertIntro();
210                         //show me!   Commented out by ART on 11-16
211                         super.show();
212                 this.shownBefore = true;
213                 }
214         }
215         else if (expertType == this.EDITOR){
216                 this.trucks         = data.getAllTrucks();
217             this.responsesDB    = data.responsesDB;
218             this.topLayerDB = data.editor_topLayerDB; //added by xing 3/13/98
219 
220             //stomp the lower level response set by filtering the
221             //the top layer responses down.
222             this.topLayerDB.updateInnerResponsesDB();
223 
224             //deal with returning users
225             if (shownBefore){
226                 showAgain();
227             }
228             else{
229                 ////create and show everything for the first time
230 
231                         //make the logic expert
232                         logicExpert = new LogicExpert();
233 
234                         //make the questions panel (added! this may be a problem)
235                         questionsPanel = new QuestionsPanel(this, editorQuestionPanelArray);
236                         this.add(questionsPanel);
237 
238                         //make the history panel
239                     expertHistory = data.editor_expertHistory;
240 
241                         //make the intro panel
242                         EditorExpertIntroPanel expertIntroPanel = new EditorExpertIntroPanel(this);
243                         this.add(expertIntroPanel);
244 
245                         //show the intro panel
246                         //really different for every expert
247                         expertIntroPanel.showExpertIntro();
248                         //show me!   Commented out by ART on 11-16
249                         super.show();
250                 this.shownBefore = true;
251                 }
252         }
253         }
254 
255         private void showAgain(){
256             //welcome the person back to the showroom
257                 if (DEBUG){System.out.println("showing the questions panel again");}
258                 makeEveryPanelInvisible();
259             ExpertReturningPanel expertReturningPanel;
260         expertReturningPanel = new ExpertReturningPanel(this);
261                 this.add(expertReturningPanel);
262             expertReturningPanel.showExpertReturn();
263             super.show();
264         }
265 
266         protected void showQuestionsPanel(){
267                 //make everything invisble I may want to
268                 makeEveryPanelInvisible();
269                 //show the questions panel with the right background
270                 questionsPanel.showQuestions(expertBackground);
271                 if (DEBUG){System.out.println("showing the questions panel");}
272         }
273 
274         protected void showNeedMoreAnswers(){
```

```
275                 makeEveryPanelInvisible();
276                 //make it
277                 NeedMoreAnswersPanel needMoreAnswersPanel;
278                 needMoreAnswersPanel = new NeedMoreAnswersPanel(this);
279                 //show the need more info panel
280                 this.add(needMoreAnswersPanel);
281                 needMoreAnswersPanel.reshape(0,0,800,600);
282                 needMoreAnswersPanel.showPanel();
283         }
284
285         public void finishedQuestions(int state){
286             this.state = state;
287                 //holy shit, the person is now done with the questions
288                 makeEveryPanelInvisible();
289
290                 //!!!!!!!!!! temporarily out so my GUI will work
291                 //this.trucks = logicExpert.makeBestBayesianGuess(trucks);
292
293                 //say goodbye to the user.
294                 //make and add the goodbye panel
295             ExpertExitPanel expertExitPanel;
296         expertExitPanel = new ExpertExitPanel(this);
297         expertExitPanel.showPanel();
298         this.add(expertExitPanel);
299         }
300
301         //should be called just from the exit panel
302         protected void leaveExpert(int state){
303             this.state = state;
304             //hide the expert panel and give the state back to the trucktown frame
305             //some last minute cleanup
306                 makeEveryPanelInvisible();
307
308             this.hide();
309             parentFrame.finishedWithModule(this, state);
310         }
311
312     //currently not needed
313         public void makeEveryPanelInvisible(){
314                 //hide everything to be able to show things in the correct order
315                 //I may want to hide all of the components in this
316                 Component[] components = this.getComponents();
317                 for (int i = 0; i<components.length; i++){
318                     components[i].hide();
319                 }
320         }
321
322     // added by Irene Wilson 2/16/98
323     // so, this violates all sorts of nebulous yet important programming principles.
324     // I really feel kinda bad about it.  But oh well.
325     public Truck[] getTrucks()
326     {
327         return trucks;
328     }
329         // done with Irene's additions
330
331         //paint methods
332         public void update (Graphics g){
333                 paint (g);
334         }
335
336         public void paint (Graphics g){
337                 g.drawImage(this.expertBackground, 0, 0, 800, 600, this);
338         }
339 }
```

```
91                  reason1 = new TextArea((""+features[0]), 4, 56, TextArea.SCROLLBARS_NONE);
92                      reason1.setBackground(Color.gray);
93                  reason1.setBounds(24,60,288,72);
94                  reason1.setFont(reasonFont);
95                  reason1.setEditable(false);
96                  add(reason1);
97          }
98                  if( (features.length > 1) && (features[1] != null) ) {
99                  //make the second reason
100                         reason2 = new TextArea((""+features[1]), 4, 56, TextArea.SCROLLBARS_NONE);
101                     reason2.setBackground(Color.gray);
102                         reason2.setBounds(24,132,288,72);
103                         reason2.setFont(reasonFont);
104                         reason2.setEditable(false);
105                         add(reason2);
106         }
107
108             if( (features.length > 2) && (features[2] != null)) {
109                 //make the third reason
110                         reason3 = new TextArea((""+features[2]), 4, 56, TextArea.SCROLLBARS_NONE);
111                     reason3.setBackground(Color.gray);
112                         reason3.setBounds(24,204,288,72);
113                         reason3.setFont(reasonFont);
114                     reason3.setEditable(false);
115                         add(reason3);
116             }
117             repaint();
118         }
119
120         public void update (Graphics g){
121                 paint (g);
122         }
123
124         public void paint (Graphics g){
125         }
126 }
```

```
1 package Trucktown.Expert;
2
3 import Trucktown.*;
4
5 import java.awt.*;
6
7 import symantec.itools.awt.WrappingLabel;
8 public class MechanicExpertIntroPanel extends Panel{
9
10         private ExpertPanel expertPanel;
11         java.awt.Button TellMoreButton;
12         symantec.itools.awt.WrappingLabel wrappingLabel1;
13         private Trucktown.ImageButton leaveButton;
14
15         //text that cycles through the expert
16         private String[] textArray =
17             {"Hello, My name is Craig Lynch.  I am the owner of Hillside Garage.  We specialize in re
   pairing trucks and sport utility vehicles.",
18             "I became a Trucktown advisor because so many of my customers have explained how challeng
   ing it can be to buy a new truck.  With so many different brands and so many different features, it c
   an be confusing.  With over 20 years of experience fixing all kind of trucks, I hope that I can help
   you find the best truck.",
19             "IBM Trucktown pays me a monthly salary for providing expert advice.  I want to stress th
   e fact that I do not receive any money from truck manufacturers.  I became a shopping expert to help
   people find the right truck, not to promote a specific brand or model.",
20             "Over the years, I have worked on just about every brand and model of trucks.  The only t
   hing is that most of my customers drive American made trucks.  I have seen most of the imports but re
   ally don't know that much about them.",
21             "I understand you are looking for a truck.  I'm going to ask you a series of questions an
   d you can end our conversation at any time."};
22         private int onString = 0;
23
24
25         //constructor
26         public MechanicExpertIntroPanel(ExpertPanel expertPanel){
27                 this.expertPanel = expertPanel;
28         }
29
30         //show it and do it!!
31         public void showExpertIntro(){
32                 setLayout(null);
33                 setBounds(0,0, 800, 600);
34                 TellMoreButton = new java.awt.Button();
35                 TellMoreButton.setLabel("Tell Me More");
36                 TellMoreButton.setBounds(342,276,168,36);
37                 TellMoreButton.setFont(expertPanel.smallFont);
38                 TellMoreButton.setBackground(expertPanel.buttonColor);
39                 add(TellMoreButton);
40                 wrappingLabel1 = new symantec.itools.awt.WrappingLabel();
41                 try {
42                         wrappingLabel1.setText(textArray[onString]);
43                         wrappingLabel1.setAlignStyle(symantec.itools.awt.WrappingLabel.ALIGN_CENTERED
   );
44                         wrappingLabel1.setBackground(Color.lightGray);
45                 }
46                 catch(java.beans.PropertyVetoException e) { }
47                 //move the on string index up
48                 onString++;
49                 wrappingLabel1.setBounds(250,50,530,190);
50                 wrappingLabel1.setFont(expertPanel.bigFont);
51                 add(wrappingLabel1);
52
53                 Image leaveImage = expertPanel.parentFrame.util.getImage( "Images/Icons/Backto.gif" )
   ;
54                 leaveButton = new ImageButton(leaveImage);
55                 leaveButton.reshape(582, 276, 75, 75);
56                 this.add(leaveButton);
57         }
58
59         //action method to deal with user hitting one of the truck image buttons
60         public boolean action(Event evt, Object arg){
61                 //dealing with a button being hit.
62                 if (TellMoreButton.equals(evt.target) ){
63                     //go through all of the text
64                     if(textArray.length > onString){
65                         try {
66                                 wrappingLabel1.setText(textArray[onString]);
67                                 wrappingLabel1.setAlignStyle(symantec.itools.awt.WrappingLabel.ALIGN_
   CENTERED);
68                         }
69                         catch(java.beans.PropertyVetoException e) { }
70                         onString++;
71                     }
72                     else {
73                     expertPanel.remove(this);
74                         expertPanel.showQuestionsPanel();
75                         }
76                         return true;
77                 }
78                 else if (leaveButton.equals(evt.target) ){
79                 //remove this and leave!
80                     expertPanel.remove(this);
```

```
1 package Trucktown.Expert;
2 import Trucktown.*;
3 import java.awt.*;
4
5 /*  same questions panel for all of the experts     */
6
7 public class QuestionsPanel extends Panel{
8
9         private ExpertPanel expertPanel;
10
11        //in the begining of life, we start with being on question zero
12        //  which is not a question but a place where we start.
13        private int questionPanelIndex = 0;
14
15        //to be shown inside the questions panel
16        private Trucktown.ImageButton previousQuestionButton;
17        public Trucktown.ImageButton doneWithQuestionsButton;
18        private Trucktown.ImageButton whatImThinkingButton;
19
20        private Image whatImThinkingImage;
21        private Image whatImThinkingNowImage;
22
23        private Truck[] whatImThinkingTrucks;
24
25    //array of question Panel names- gotten when the thing is constructed
26    //abstracted out to have the class calling the constructor to have control over the question ord
   er
27    //sort of silly cuz this will soon be handled by the logic expert
28    private String[] questionPanelArray;
29
30        //deal with returning users
31        private boolean shownQuestionsBefore = false;
32
33    private Image backgroundImage;
34
35        //constructor
36        public QuestionsPanel(ExpertPanel expertPanel, String[] questionPanelArray){
37                this.expertPanel = expertPanel;
38                this.questionPanelArray = questionPanelArray;
39        }
40
41        //show it and do it!!
42        public void showQuestions(Image backgroundImage){
43            this.backgroundImage = backgroundImage;
44
45        //deal with returning users
46        if (shownQuestionsBefore){
47            showAgain();
48        }
49        else{
50            shownQuestionsBefore = true;
51                this.setLayout(null);
52
53                //note: if more buttons are added, then change clean panel!!!
54                //make the previous question button
55                Image previousQuestionImage = expertPanel.parentFrame.util.getImage( "Images/Icons/b
   ack.gif" );
56                previousQuestionButton = new ImageButton(previousQuestionImage);
57                previousQuestionButton.reshape(450, 495, 75, 75);
58                this.add(previousQuestionButton);
59                //make the DoneWithQuestionsButton
60                Image doneWithQuestionsImage = expertPanel.parentFrame.util.getImage( "Images/Icons/
   backto.gif" );
61                doneWithQuestionsButton = new ImageButton(doneWithQuestionsImage);
62                doneWithQuestionsButton.reshape(575,495, 75, 75);
63                this.add(doneWithQuestionsButton);
64
65
66                //make the what Im thinking Button
67                whatImThinkingImage = expertPanel.parentFrame.util.getImage( "Images/Icons/think.gif
   " );
68                whatImThinkingNowImage = expertPanel.parentFrame.util.getImage( "Images/Icons/thinki
   ngAnimation.gif" );
69                whatImThinkingButton = new ImageButton(whatImThinkingImage);
70                whatImThinkingButton.reshape(25,25, 75, 75);
71                whatImThinkingButton.hide();
72                this.add(whatImThinkingButton);
73
74                //show the first question by asking the next question
75                showNextQuestion();
76
77                this.reshape(0,0,800,600);
78                this.show();
79                }
80        }
81
82        private void showAgain(){
83                this.show();
84                //make sure that the buttons are showing
85                        doneWithQuestionsButton.show();
86                previousQuestionButton.show();
87                whatImThinkingButton.hide();
```

```
88                  //just show the question that we left off on
89                  this.showQuestion();
90          }
91
92          //action method to deal with user hitting one of the truck image buttons
93          public boolean action(Event evt, Object arg){
94                  //dealing with a button being hit.
95                  if (previousQuestionButton.equals(evt.target) ){
96                          //make sure that the done button is now showing(cuz we may have left from th
    e last screen)
97                          doneWithQuestionsButton.show();
98                          showPreviousQuestion();
99                          return true;
100                 }
101                 else if (doneWithQuestionsButton.equals(evt.target) ){
102                         userIsDone();
103                         return true;
104                 }
105                 else if (whatImThinkingButton.equals(evt.target) ){
106                     showWhatImThinkingPanel();
107                         return true;
108                 }
109                 //
110                 //
111                 //help buttons implemented by question panels
112                 //
113                 //
114                 else if (arg.equals("Why some people like compact trucks")){
115                     //implemented by question 1
116                 Image helpImage = expertPanel.parentFrame.util.getImage( "Images/Help/like_compact.j
    pg" );
117                         //hide the questions panel
118                         this.expertPanel.makeEveryPanelInvisible();
119                 HelpPanel helpPanel = new HelpPanel(expertPanel);
120                 this.expertPanel.add(helpPanel);
121             helpPanel.showHelp(helpImage);
122                         return true;
123                 }
124                 else if (arg.equals("Why some people like full size trucks")){
125                     //implemented by question 1
126                 Image helpImage = expertPanel.parentFrame.util.getImage( "Images/Help/like_full.jpg"
    );
127                         //hide the questions panel
128                         this.expertPanel.makeEveryPanelInvisible();
129                 HelpPanel helpPanel = new HelpPanel(expertPanel);
130                 this.expertPanel.add(helpPanel);
131             helpPanel.showHelp(helpImage);
132                         return true;
133                 }
134                 else if (arg.equals("Trailer Weights")){
135                     //implemented by question 3
136                 Image helpImage = expertPanel.parentFrame.util.getImage( "Images/Help/trailer_wt.jpg
    " );
137                         //hide the questions panel
138                         this.expertPanel.makeEveryPanelInvisible();
139                 HelpPanel helpPanel = new HelpPanel(expertPanel);
140                 this.expertPanel.add(helpPanel);
141             helpPanel.showHelp(helpImage);
142                         return true;
143                 }
144                 else if (arg.equals("Truck prices today")){
145                     //implemented by question 5
146                 Image helpImage = expertPanel.parentFrame.util.getImage( "Images/Help/prices_today.j
    pg" );
147                         //hide the questions panel
148                         this.expertPanel.makeEveryPanelInvisible();
149                 HelpPanel helpPanel = new HelpPanel(expertPanel);
150                 this.expertPanel.add(helpPanel);
151             helpPanel.showHelp(helpImage);
152                         return true;
153                 }
154                 else if (arg.equals("Why some people like short bed trucks")){
155                     //implemented by question 11
156                 Image helpImage = expertPanel.parentFrame.util.getImage( "Images/Help/like_short.jpg
    " );
157                         //hide the questions panel
158                         this.expertPanel.makeEveryPanelInvisible();
159                 HelpPanel helpPanel = new HelpPanel(expertPanel);
160                 this.expertPanel.add(helpPanel);
161             helpPanel.showHelp(helpImage);
162                         return true;
163                 }
164                 else if (arg.equals("Why some people like long bed trucks")){
165                     //implemented by question 11
166                 Image helpImage = expertPanel.parentFrame.util.getImage( "Images/Help/like_long.jpg"
    );
167                         //hide the questions panel
168                         this.expertPanel.makeEveryPanelInvisible();
169                 HelpPanel helpPanel = new HelpPanel(expertPanel);
170                 this.expertPanel.add(helpPanel);
171             helpPanel.showHelp(helpImage);
172                         return true;
```

```
173                 }
174                 else if (arg.equals("Why some people like smaller engines")){
175                     //implemented by question 3
176                 Image helpImage = expertPanel.parentFrame.util.getImage( "Images/Help/like_v46.jpg"
    );
177                         //hide the questions panel
178                         this.expertPanel.makeEveryPanelInvisible();
179                 HelpPanel helpPanel = new HelpPanel(expertPanel);
180                 this.expertPanel.add(helpPanel);
181             helpPanel.showHelp(helpImage);
182                         return true;
183                 }
184                 else if (arg.equals("Why some people like larger engines")){
185                     //implemented by question 3
186                 Image helpImage = expertPanel.parentFrame.util.getImage( "Images/Help/like_v8.jpg" )
    ;
187                         //hide the questions panel
188                         this.expertPanel.makeEveryPanelInvisible();
189                 HelpPanel helpPanel = new HelpPanel(expertPanel);
190                 this.expertPanel.add(helpPanel);
191             helpPanel.showHelp(helpImage);
192                         return true;
193                 }
194                 else if (arg.equals("What's the difference")){
195                     //implemented by question 3
196                 Image helpImage = expertPanel.parentFrame.util.getImage( "Images/Help/diff_future.jp
    g" );
197                         //hide the questions panel
198                         this.expertPanel.makeEveryPanelInvisible();
199                 HelpPanel helpPanel = new HelpPanel(expertPanel);
200                 this.expertPanel.add(helpPanel);
201             helpPanel.showHelp(helpImage);
202                         return true;
203                 }
204
205                 return false;
206         }
207
208     //show methods
209     public void showWhatImThinkingPanel(){
210
211                 //hide the questions panel
212             this.expertPanel.makeEveryPanelInvisible();
213                 //make the what i'm thinking panel
214         WhatImThinkingPanel whatImThinkingPanel = new WhatImThinkingPanel(expertPanel);
215
216                 this.expertPanel.add(whatImThinkingPanel);
217             whatImThinkingPanel.showWhatImThinking(expertPanel.parentFrame.util.getImage( "Images/cl
    ouds.jpg" ));
218         }
219
220         public void showNextQuestion(){
221         /* this routine may be replaced by a more generic beast that asks a question expert what the
     next question it
222         should ask should be.  it will probably then add to a questionsasked array to remember the o
    rder that I've asked
223         */
224
225                 //increments the question index
226                 this.questionPanelIndex++;
227                 showQuestion(this.questionPanelIndex);
228         }
229
230         private void showPreviousQuestion(){
231                 //decrements question index if it is greater than 1
232                 if (1 < this.questionPanelIndex){
233                         //back up
234                         this.questionPanelIndex--;
235                 }
236                 else {
237                         //show the first question
238                         this.questionPanelIndex = 1;
239                 }
240                 showQuestion(this.questionPanelIndex);
241         }
242
243         public void showQuestion(){
244             //show what ever was shown last
245                 showQuestion(this.questionPanelIndex);
246         }
247
248         private void showQuestion(int questionPanelIndex){
249                 /* the questionpanelarray may become the history panel array that is constructed as
    the user moves
250                 forward through the questionnaire
251                 */
252
253                 if (expertPanel.DEBUG){
254                                 System.out.println("question Index in showQuestion:"+questionPanelIn
    dex);
255                 }
256
```

```
257                 //remove everything but the buttons
258                 cleanPanel();
259
260                 //think about what the user has said and update the logic
261                 //axed out, and strip out if the expert's logic is doing just fine
262                 //expertPanel.trucks = expertPanel.logicExpert.makeBestGuess(expertPanel.trucks, exp
    ertPanel.responsesDB);
263
264
265         //deal with the what i am thinking button.
266         Truck[] newWhatImThinkingTrucks = expertPanel.logicExpert.get4RecommendedBayesianAndSegmentT
    rucks(expertPanel.responsesDB, expertPanel.trucks);
267
268         //show the button and tell the history object if the user has some trucks to show
269         if (newWhatImThinkingTrucks[0] != null){
270             //there are trucks there!
271             whatImThinkingButton.show();
272             expertPanel.expertHistory.expertIsThinkingSomething = true;
273             //this is the point at which the showroom can use this expert's information
274             expertPanel.data.lastExpertRecommender = expertPanel.expertType;
275         }
276         else {
277             //not confident enough
278             whatImThinkingButton.hide();
279             expertPanel.expertHistory.expertIsThinkingSomething = false;
280         }
281         //show the correct image
282         //poor code sorry
283         if (whatImThinkingTrucks != null){
284             if (whatImThinkingTrucks[0] == newWhatImThinkingTrucks[0] &&
285                 whatImThinkingTrucks[1] == newWhatImThinkingTrucks[1] &&
286                 whatImThinkingTrucks[2] == newWhatImThinkingTrucks[2] &&
287                 whatImThinkingTrucks[3] == newWhatImThinkingTrucks[3]){
288                 //nothing has changed
289                     whatImThinkingButton.setUnarmedImage(whatImThinkingImage);
290
291             }
292             else{
293                     //show the NEW button cuz the trucks have changed
294                     whatImThinkingButton.setUnarmedImage(whatImThinkingNowImage);
295                 }
296         }
297         else{
298             //show the NEW button cuz the trucks have changed
299             whatImThinkingButton.setUnarmedImage(whatImThinkingNowImage);
300         }
301         whatImThinkingTrucks = newWhatImThinkingTrucks;
302
303                 //deal with making the status frame do something
304         expertPanel.statusFrame.updateStatus(expertPanel.responsesDB, expertPanel.trucks);

305
306                 //try to make sure questionPanelIndex is within the bounds of the questionpanel arra
    y
307                 try {
308                     try {
309                     //hard core programming that takes the name in the question panel array,
310                     //makes an object out of it, adds it,  and shows it
311                     Class QPanel = Class.forName(questionPanelArray[questionPanelIndex]);
312                     Class params[] = new Class[1];
313                     params[0] = expertPanel.getClass();
314                     java.lang.reflect.Constructor constructor = QPanel.getConstructor(params);
315                     Object objParams[] = new Object[1];
316                     objParams[0] = expertPanel;
317                     Object questionPanel = constructor.newInstance(objParams);
318                     this.add((Component)questionPanel);
319                     ((QuestionInterface)questionPanel).showQuestion();
320
321             }
322                     catch(Exception classE){
323                     //must be done because there was no class behind the string name in the question
    panel array
324                         if (expertPanel.DEBUG){
325
326                                     System.out.println(classE);
327                         }
328
329                         DoneWithQuestions doneWithQuestions = new DoneWithQuestions(expertPanel);
330                         doneWithQuestions.showQuestion_Configuration();
331                 }
332         }
333                 catch(ArrayIndexOutOfBoundsException e){
334                         //ran through all of the questions, must be done
335                 if (expertPanel.DEBUG){
336                                 System.out.println("finished with question array");
337                 }
338                 DoneWithQuestions doneWithQuestions = new DoneWithQuestions(expertPanel);
339                         doneWithQuestions.showQuestion_Configuration();
340                 }
341         }
342
343         public void cleanPanel(){
```

```
344                 //remove everything added to the panel (exept for the first 3 components
345                 //  wich are ubiquitous buttons)
346                 doneWithQuestionsButton.show(); //just make sure that its showing
347                 try {
348                         Component component;
349
350                 int i = 3;
351                         while(true){
352                                 component = this.getComponent(i);
353                                 //debug
354                 if (expertPanel.DEBUG){
355                                 System.out.println(i+"=i  and component:"+component);
356                 }
357                             this.remove(component);
358                                 component = null;
359                         }
360                 }
361                 catch(ArrayIndexOutOfBoundsException e){
362                 if (expertPanel.DEBUG){
363                                 System.out.println("hit the end of the array");
364                 }
365                         //hit one after the end of the component array
366                         //do nothing and continue with life
367                 }
368         }
369
370         public void finishedQuestion(){
371                 ///the user is done with the question
372                 if (expertPanel.DEBUG){
373                         System.out.println("question Index of the panel just finished:"+questionPane
    lIndex);
374                         expertPanel.responsesDB.print();
375                 }
376
377                 //show next question
378             showNextQuestion();
379         }
380
381         private void userIsDone(){
382
383                 //calls the method in the done class
384                 //make the DoneWithQuestions class
385                 DoneWithQuestions doneWithQuestions = new DoneWithQuestions(expertPanel);
386                 doneWithQuestions.done();
387         }
388
389         //paint methods
390         //update or paint method here!!
391         //paint methods
392         public void update (Graphics g){
393                 paint (g);
394         }
395
396         public void paint (Graphics g){
397                 g.drawImage(this.backgroundImage, 0, 0, 800, 600, this);
398         }
399 }
```

```
1 package Trucktown.Expert;
2
3 import Trucktown.*;
4
5 import java.awt.*;
6
7 import symantec.itools.awt.WrappingLabel;
8 public class Question_1Panel extends Panel implements QuestionInterface{
9
10         private ExpertPanel expertPanel;
11     //private Response[] respArray;
12
13
14     TopLayerResponseDB topLayerDB;
15     //added by xing 3/13/98
16
17         java.awt.Button button2;
18         java.awt.Button button1;
19         symantec.itools.awt.WrappingLabel wrappingLabel1;
20         symantec.itools.awt.WrappingLabel wrappingLabel2;
21     java.awt.Button button3;
22
23
24         //constructor
25         public Question_1Panel(ExpertPanel expertPanel){
26                 this.expertPanel = expertPanel;
27         }
28
29         //show it and do it!!
30         public void showQuestion(){
31
32                 System.out.println("showing question 1");
33
34                 setLayout(null);
35                 setBounds(380, 50, 378, 295);
36                 setBackground(Color.lightGray);
37
38                 //respArray = expertPanel.responsesDB.responseArray;
39         topLayerDB = expertPanel.topLayerDB;
40
41         button2 = new java.awt.Button();
42                 button2.setLabel("Only full size trucks");
43                 button2.setBounds(105,140,170,36);
44                 button2.setFont(expertPanel.smallFont);
45                 button2.setBackground(expertPanel.buttonColor);
46                 add(button2);
47                 if(topLayerDB.getTLVal(TopLayerResponseDB.FULLORCOMPACT_TOP) == 2) button2.requestFo
   cus();
48                 button1 = new java.awt.Button();
49                 button1.setLabel("Only compact trucks");
50                 button1.setBounds(105,190,170,36);
51                 button1.setFont(expertPanel.smallFont);
52                 button1.setBackground(expertPanel.buttonColor);
53                 add(button1);
54                 if(topLayerDB.getTLVal(TopLayerResponseDB.FULLORCOMPACT_TOP) == 1) button1.requestFo
   cus();
55                 wrappingLabel1 = new symantec.itools.awt.WrappingLabel();
56                 wrappingLabel2 = new symantec.itools.awt.WrappingLabel();
57                 try
58                 {
59                         wrappingLabel1.setText("Trucks basically come in two sizes- full size and co
   mpact.");
60
61                     wrappingLabel2.setText("Do you have a certain size truck in mind?");
62                 }
63                 catch(java.beans.PropertyVetoException e) { }
64                 try
65                 {
66                         wrappingLabel1.setAlignStyle(symantec.itools.awt.WrappingLabel.ALIGN_CENTERE
   D);
67                     wrappingLabel2.setAlignStyle(symantec.itools.awt.WrappingLabel.ALIGN_CENTERED);
68                 }
69                 catch(java.beans.PropertyVetoException e) { }
70                 wrappingLabel1.setBounds(9,5,366,55);
71                 wrappingLabel1.setFont(expertPanel.bigFont);
72                 add(wrappingLabel1);
73                 wrappingLabel2.setBounds(9,80,366,55);
74                 wrappingLabel2.setFont(expertPanel.bigFont);
75                 add(wrappingLabel2);
76                 button3 = new java.awt.Button();
77                 button3.setLabel("Either size is OK");
78                 button3.setBounds(105,240,170,36);
79                 button3.setFont(expertPanel.smallFont);
80                 button3.setBackground(expertPanel.buttonColor);
81                 add(button3);
82                 if(topLayerDB.getTLVal(TopLayerResponseDB.FULLORCOMPACT_TOP) == 3) button3.requestFo
   cus();
83                 //changed by xing 3/13/98
84                 //adding a help button in the questionspanel
85                 Button helpButton = new java.awt.Button();
86                 helpButton.setLabel("Why some people like compact trucks");
87                 helpButton.setBounds(getLocation().x + 31 ,420,316,36);
```

```
88                  helpButton.setFont(expertPanel.smallFont);
89                  helpButton.setBackground(expertPanel.helpButColor);
90                  expertPanel.questionsPanel.add(helpButton);
91
92                  Button help2Button = new java.awt.Button();
93                  help2Button.setLabel("Why some people like full size trucks");
94                  help2Button.setBounds(getLocation().x + 31,370,316,36);
95                  help2Button.setFont(expertPanel.smallFont);
96                  help2Button.setBackground(expertPanel.helpButColor);
97                  expertPanel.questionsPanel.add(help2Button);
98
99          }
100
101         //action method to deal with user hitting one of the truck image buttons
102         public boolean action(Event evt, Object arg){
103                 //dealing with a button being hit.
104                 if (button1.equals(evt.target) ){
105                         response(1);
106                         return true;
107                 }
108                 else if (button2.equals(evt.target) ){
109                         response(2);
110                         return true;
111                 }
112                 else if (button3.equals(evt.target) ){
113                         response(3);
114                         return true;
115                 }
116                 return false;
117         }
118
119         public void response(int response){
120                 //enter into the responses db what the response was
121                 topLayerDB.setTLVal(TopLayerResponseDB.FULLORCOMPACT_TOP, response);
122         //modified by Xing 3/13/98
123                 //tell the questions panel that this question is finished
124                 expertPanel.questionsPanel.finishedQuestion();
125         }
126
127         //paint methods
128         //update or paint method here!!
129         //paint methods
130         public void update (Graphics g){
131                 paint (g);
132         }
133
134         public void paint (Graphics g){
135         }
136 }
137
```

```
1 package Trucktown.Expert;
2
3 public interface QuestionInterface{
4
5     public abstract void showQuestion();
6
7     public abstract void reshape(int x, int y, int h, int w);
8
9 }
```

```
1 package Trucktown.Expert;
2
3 import Trucktown.*;
4
5 //
6 //
7 // DoneWithQuestions
8 //
9 /* this little beast decides what to do when the user is done with the regular questions
10 */
11 //
12 public class DoneWithQuestions {
13         private ExpertPanel expertPanel;
14
15         //constructor
16         public DoneWithQuestions(ExpertPanel expertPanel){
17                 this.expertPanel = expertPanel;
18         }
19
20         ///methods
21         public void done(){
22                 //see if enough questions were answered to continue
23                 //on to the config question
24                 if ((expertPanel.expertHistory.expertIsThinkingSomething) &&
25                     (expertPanel.expertHistory.questionConfigPanelShown == false)){
26                         showQuestion_Configuration();
27                 }
28                 else if ((expertPanel.expertHistory.expertIsThinkingSomething) &&
29                     (expertPanel.expertHistory.questionConfigPanelShown == true)){
30             //just be done so skip it and get the hell out of here
31                         expertPanel.finishedQuestions(ModuleState.DONE_VISIT);

32         }
33                 else{
34                         showNeedMoreInfo();
35                 }
36         }
37
38         private void showNeedMoreInfo(){
39                 //make it
40                 NeedMoreAnswersPanel needMoreAnswersPanel;
41                 needMoreAnswersPanel = new NeedMoreAnswersPanel(expertPanel);
42                 //show the need more info panel
43                 expertPanel.questionsPanel.doneWithQuestionsButton.hide();
44                 expertPanel.questionsPanel.cleanPanel();
45                 expertPanel.questionsPanel.add(needMoreAnswersPanel);
46                 expertPanel.questionsPanel.show();
47                 needMoreAnswersPanel.reshape(0,0,800,600);
48                 needMoreAnswersPanel.showPanel();
49                 if (expertPanel.DEBUG){System.out.println("config it");}
50         }
51
52
53         public void showQuestion_Configuration(){
54                 //show it cuz it hasn't been shown before
55                 Question_ConfigurationPanel question_ConfigurationPanel;
56                 question_ConfigurationPanel = new Question_ConfigurationPanel(expertPanel);
57                 //show the config screen and hiding the done button
58                 expertPanel.questionsPanel.doneWithQuestionsButton.hide();
59                 expertPanel.questionsPanel.cleanPanel();
60                 expertPanel.questionsPanel.add(question_ConfigurationPanel);
61                 question_ConfigurationPanel.reshape(0,0,800,700);
62                 question_ConfigurationPanel.showQuestion_ConfigurationPanel();
63         }
64 }
65
```

```
1 package Trucktown.Expert;
2
3 import Trucktown.*;
4
5 import java.awt.*;
6
7 import symantec.itools.awt.WrappingLabel;
8 public class NeedMoreAnswersPanel extends Panel{
9
10         private ExpertPanel expertPanel;
11
12         symantec.itools.awt.WrappingLabel wrappingLabel1;
13         private Trucktown.ImageButton backToMapButton;
14
15         //constructor
16         public NeedMoreAnswersPanel(ExpertPanel expertPanel){
17                 this.expertPanel = expertPanel;
18         }
19
20         //show it and do it!!
21         public void showPanel(){
22
23                 setLayout(null);
24                 setBounds(300, 80, 410, 270);
25                 setBackground(Color.lightGray);
26                 wrappingLabel1 = new symantec.itools.awt.WrappingLabel();
27                 try {
28                         wrappingLabel1.setText("Sorry that you couldn't stay to finish.  I suggest th
   at you choose another advisor or visit automile.");
29                         wrappingLabel1.setAlignStyle(symantec.itools.awt.WrappingLabel.ALIGN_CENTERED
   );
30                 }
31                 catch(java.beans.PropertyVetoException e) { }
32                 wrappingLabel1.setBounds(12,12,384,124);
33                 wrappingLabel1.setFont(expertPanel.bigFont);
34                 add(wrappingLabel1);
35                 //make the DoneWithQuestionsButton
36                 Image doneWithQuestionsImage = expertPanel.parentFrame.util.getImage( "Images/Expert/
   back_map.jpg" );
37                 backToMapButton = new ImageButton(doneWithQuestionsImage);
38                 this.add(backToMapButton);
39                 backToMapButton.reshape(135, 140,115, 85);
40
41                 expertPanel.questionsPanel.doneWithQuestionsButton.hide();
42
43         }
44
45         //action method to deal with user
46         public boolean action(Event evt, Object arg){
47                 //dealing with a button being hit.
48                 if (backToMapButton.equals(evt.target) ){
49                 //remove this and leave!
50                 this.getParent().remove(this);
51             expertPanel.leaveExpert(ModuleState.PARTIAL_VISIT);
52                         return true;
53                 }
54
55                 return false;
56         }
57
58         //paint methods
59         //update or paint method here!!
60         //paint methods
61         public void update (Graphics g){
62                 paint (g);
63         }
64
65         public void paint (Graphics g){
66         }
67 }
68
```

```
1  package Trucktown.Expert;
2
3  import java.awt.*;
4  import Trucktown.*;
5
6  public class StatusFrame extends Frame{
7
8          ExpertPanel expertPanel;
9          private Truck[] trucks;
10         //private int[] segmentArray;
11         private ResponsesDB responsesDB;
12
13     private Choice truckChoice;
14         java.awt.Button updateDeltasButton;
15         private BayesianExpert bayesianExpert;
16     private boolean firstTime = true;
17
18     public StatusFrame(ExpertPanel expertPanel){
19         this.expertPanel = expertPanel;
20         this.setBounds(700,0, 700, 600);
21
22         super.show();
23     }
24
25     public void updateStatus(ResponsesDB responsesDB, Truck[] trucks){
26         this.responsesDB = responsesDB;
27         this.trucks = trucks;
28         //this.segmentArray = segmentArray;
29
30         if (firstTime){
31             /*
32                 updateDeltasButton = new java.awt.Button();
33                 updateDeltasButton.setLabel("OK");
34                 updateDeltasButton.setBounds(110,200,170,36);
35                 updateDeltasButton.setFont(expertPanel.smallFont);
36                 updateDeltasButton.setBackground(expertPanel.buttonColor);
37                 add(updateDeltasButton);
38             */
39             //to able to choose the truck to view it's response deltas
40             /*
41             truckChoice = new Choice();
42             truckChoice.setBounds(70,144,250,36);
43             for(int i = 0; i < trucks.length; i++)
44             {
45                 truckChoice.add(trucks[i].GetString(trucks[1].VEHICLE_STR));
46             }
47             add(truckChoice);
48             */
49             firstTime = false;
50         }
51
52         this.update(this.getGraphics());
53     }
54
55     //show the deltas of questions for the given truck in the pull down menu
56         public boolean action(Event evt, Object arg)
57         {
58                 //dealing with a button being hit.
59                 if (updateDeltasButton.equals(evt.target) ){
60                         // update the deltas for this truck
61
62             //make a bayesian expert to ask
63             BayesianExpert bayesianExpert = new BayesianExpert();
64             //ask the expert fill the responseDB with delta's
65             //Truck selectedTruck = expertPanel.data.truckDB.getTruckFromName(truckChoice.getSelecte
   dItem());
66             //this.responsesDB = bayesianExpert.makeDeltas(this.responsesDB, this.trucks, selectedTr
   uck);
67             //this.responsesDB = bayesianExpert.rankResponsesByDelta(this.responsesDB);
68
69             this.update(this.getGraphics());
70
71                         return true;
72                 }
73                 return false;
74         }
75
76
77         //paint methods
78         //update or paint method here!!
79         //paint methods
80         public void paint (Graphics g){
81
82             try{
83
84                 //print the segments the user belongs to
85                 //g.drawString("user belongs to segment:", 20, 40);
86                 //for (int segmentIndex=0; segmentIndex<segmentArray.length; segmentIndex++){
87                 //    g.drawString(String.valueOf(segmentArray[segmentIndex]), (160 + 10*segmentInde
   x), 40);
88                 //}
89
```

```
1 package Trucktown.Expert;
2
3 import Trucktown.*;
4
5 import java.awt.*;
6
7 import symantec.itools.awt.WrappingLabel;
8 public class ExpertExitPanel extends Panel{
9
10         private ExpertPanel expertPanel;
11
12         symantec.itools.awt.WrappingLabel wrappingLabel1;
13         private Trucktown.ImageButton leaveButton;
14
15         //constructor
16         public ExpertExitPanel(ExpertPanel expertPanel){
17                 this.expertPanel = expertPanel;
18         }
19
20         //show it and do it!!
21         public void showPanel(){
22
23                 setLayout(null);
24                 setBounds(300, 80, 410, 232);
25                 setBackground(Color.lightGray);
26
27                 wrappingLabel1 = new symantec.itools.awt.WrappingLabel();
28                 try {
29                         wrappingLabel1.setText("I think that you have told me enough to tell the peop
   le at the showroom what to show you.  Thank's for stopping by!");
30                         wrappingLabel1.setAlignStyle(symantec.itools.awt.WrappingLabel.ALIGN_CENTERED
   );
31                 }
32                 catch(java.beans.PropertyVetoException e) { }
33                 wrappingLabel1.setBounds(12,12,384,84);
34                 wrappingLabel1.setFont(expertPanel.bigFont);
35                 add(wrappingLabel1);
36
37                 Image leaveImage = expertPanel.parentFrame.util.getImage( "Images/Expert/back_map.jpg
   " );
38                 leaveButton = new ImageButton(leaveImage);
39                 leaveButton.reshape(142, 108, 115, 85);
40                 this.add(leaveButton);
41         }
42
43         //action method to deal with user
44         public boolean action(Event evt, Object arg){
45                 //dealing with a button being hit.
46                 if (leaveButton.equals(evt.target) ){
47                 //remove this and leave!
48                 expertPanel.remove(this);
49             expertPanel.leaveExpert(ModuleState.DONE_VISIT);
50
51                         return true;
52                 }
53                 return false;
54         }
55
56         //paint methods
57         //update or paint method here!!
58         //paint methods
59         public void update (Graphics g){
60                 paint (g);
61         }
62
63         public void paint (Graphics g){
64         }
65 }
66
```

```
1 package Trucktown.Expert;
2
3 import Trucktown.*;
4
5 import java.awt.*;
6 import symantec.itools.awt.WrappingLabel;
7
8 public class ExpertReturningPanel extends Panel{
9
10         private ExpertPanel expertPanel;
11         symantec.itools.awt.WrappingLabel wrappingLabel1;
12         private Trucktown.ImageButton gotoConversationButton;
13
14         //constructor
15         public ExpertReturningPanel(ExpertPanel expertPanel){
16                 this.expertPanel = expertPanel;
17         }
18
19         //show it and do it!!
20         public void showExpertReturn(){
21                 setLayout(null);
22                 setBounds(300, 80, 410, 232);
23                 setBackground(Color.lightGray);
24
25                 wrappingLabel1 = new symantec.itools.awt.WrappingLabel();
26                 try {
27                         wrappingLabel1.setText("Welcome back!  Let's continue where we left off.");
28                         wrappingLabel1.setAlignStyle(symantec.itools.awt.WrappingLabel.ALIGN_CENTERED
   );
29                 }
30                 catch(java.beans.PropertyVetoException e) { }
31                 wrappingLabel1.setBounds(12,12,384,84);
32                 wrappingLabel1.setFont(expertPanel.bigFont);
33                 add(wrappingLabel1);
34
35                 Image conversationImage = expertPanel.parentFrame.util.getImage( "Images/Icons/backTo
   Conversation.gif" );
36                 gotoConversationButton = new ImageButton(conversationImage);
37                 gotoConversationButton.reshape(122, 108, 140, 70);
38                 this.add(gotoConversationButton);
39         }
40
41
42
43         //action method to deal with user hitting one of the truck image buttons
44         public boolean action(Event evt, Object arg){
45                 //dealing with a button being hit.
46                 if (gotoConversationButton.equals(evt.target) ){
47                         expertPanel.showQuestionsPanel();
48                         expertPanel.remove(this);
49                         return true;
50                 }
51
52                 return false;
53         }
54
55         //paint methods
56         //update or paint method here!!
57         //paint methods
58         public void update (Graphics g){
59                 paint (g);
60         }
61
62         public void paint (Graphics g){
63         }
64 }
```

```
1 package Trucktown.ShowRoom;
2
3 import Trucktown.*;
4 import Trucktown.Expert.*;
5
6 import java.awt.*;
7 import java.net.*;
8 import java.applet.*;
9 import symantec.itools.awt.WrappingLabel;
10
11 public class BayesianSegmentRoom extends Panel{
12
13         private ShowRoomPanel showRoomPanel;
14         ImageButton button1;
15
16     //bayesian trucks
17         LtLabel truckLabel1;
18         LtLabel truckLabel2;
19         private Trucktown.ImageButton truckButton_1;
20         private Trucktown.ImageButton truckButton_2;
21
22         //segment trucks
23         LtLabel truckLabel3;
24         LtLabel truckLabel4;
25         LghtLabel infoLabel;
26         private Trucktown.ImageButton truckButton_3;
27         private Trucktown.ImageButton truckButton_4;
28     private LabHolder labHold = new LabHolder();
29
30         public Truck[] topTrucks;
31
32         Image backgroundImage;
33
34         //constructor
35         public BayesianSegmentRoom(ShowRoomPanel showRoomPanel)
36         {
37                 this.showRoomPanel = showRoomPanel;
38                 setLayout(null);
39                 setSize(800,600);
40
41                 LtLabel Lab1 = new LtLabel("Here are a few trucks that I think you might like.", LtL
   abel.CENTER);
42                 Lab1.setFont(showRoomPanel.bigFont);
43                 Lab1.setBounds(225, 20, 550, 35);
44                 labHold.add(Lab1);
45
46                 LtLabel Lab2 = new LtLabel("To learn more about a truck, click on its image.", LtLab
   el.CENTER);
47                 Lab2.setFont(showRoomPanel.bigFont);
48                 Lab2.setBounds(225, 55, 550, 35);
49                 labHold.add(Lab2);
50
51         //make the background clouds (should go else where)
52                 this.backgroundImage = showRoomPanel.parentFrame.util.getImage( "Images/clouds.jpg"
   );
53
54                 //make "take me to trucktown" image button
55                 Image backToTrucktownImage = showRoomPanel.parentFrame.util.getImage( "Images/Icons/
   Mainmap.gif" );
56                 button1 = new ImageButton(backToTrucktownImage);
57                 button1.reshape(50,25, 140, 70);
58                 this.add(button1);
59
60
61
62                 //figure out what four baysian trucks to show (but just show two of these
63                 this.topTrucks = getTopTrucks(showRoomPanel.responsesDB, showRoomPanel.trucks);
64                 //see if the last truck was filled in
65                 try {
66                     String s = topTrucks[3].GetString(Truck.VEHICLE_STR);
67                 }
68                 catch (NullPointerException e){
69                     //didn't get a truck so fill it up
70                     System.out.println("oops, there were no trucks from the BAYESIAN analysis");
71                     topTrucks[0] = showRoomPanel.trucks[0];
72                     topTrucks[1] = showRoomPanel.trucks[1];
73                     topTrucks[2] = showRoomPanel.trucks[2];
74                     topTrucks[3] = showRoomPanel.trucks[3];
75                 }
76
77                 //first truck
78                 truckLabel1 = new LtLabel(topTrucks[0].GetString(Truck.VEHICLE_STR), LghtLabel.CENTE
   R);
79                 truckLabel1.setBounds(105,120,250,30);
80                 truckLabel1.setFont(showRoomPanel.smallFont);
81                 labHold.add(truckLabel1);
82
83         Image truckImage_1 = showRoomPanel.parentFrame.util.getImage( topTrucks[0].GetString(Truck.I
   MAGE_STR) );
84         truckImage_1 = showRoomPanel.parentFrame.util.getScaledImage(truckImage_1, 225, 148);
85                 truckButton_1 = new ImageButton(truckImage_1);
86                 truckButton_1.reshape(110, 150, 225, 148);
```

```
87                  this.add(truckButton_1);
88
89                  //second truck
90                  truckLabel2 = new LtLabel(topTrucks[1].GetString(Truck.VEHICLE_STR), LghtLabel.CENTE
    R);
91                  truckLabel2.setBounds(435,120,250,30);
92                  truckLabel2.setFont(showRoomPanel.smallFont);
93                  labHold.add(truckLabel2);
94
95          Image truckImage_2 = showRoomPanel.parentFrame.util.getImage( topTrucks[1].GetString(Truck.I
    MAGE_STR) );
96          truckImage_2 = showRoomPanel.parentFrame.util.getScaledImage(truckImage_2, 225, 148);
97                  truckButton_2 = new ImageButton(truckImage_2);
98                  truckButton_2.reshape(440, 150, 225, 148);
99                  this.add(truckButton_2);
100
101                 //third truck
102                 truckLabel3 = new LtLabel(topTrucks[2].GetString(Truck.VEHICLE_STR), LghtLabel.CENTE
    R);
103                 truckLabel3.setBounds(105,360,250,30);
104                 truckLabel3.setFont(showRoomPanel.smallFont);
105                 labHold.add(truckLabel3);
106
107         Image truckImage_3 = showRoomPanel.parentFrame.util.getImage( topTrucks[2].GetString(Truck.I
    MAGE_STR) );
108         truckImage_3 = showRoomPanel.parentFrame.util.getScaledImage(truckImage_3, 225, 148);
109                 truckButton_3 = new ImageButton(truckImage_3);
110                 truckButton_3.reshape(110, 390, 225, 148);
111                 this.add(truckButton_3);
112
113                 //fourth truck
114                 truckLabel4 = new LtLabel(topTrucks[3].GetString(Truck.VEHICLE_STR), LghtLabel.CENTE
    R);
115                 truckLabel4.setBounds(435,360,250,30);
116                 truckLabel4.setFont(showRoomPanel.smallFont);
117                 labHold.add(truckLabel4);
118
119                 Image truckImage_4 = showRoomPanel.parentFrame.util.getImage( topTrucks[3].GetString
    (Truck.IMAGE_STR) );
120         truckImage_4 = showRoomPanel.parentFrame.util.getScaledImage(truckImage_4, 225, 148);
121                 truckButton_4 = new ImageButton(truckImage_4);
122                 truckButton_4.reshape(440, 390, 225, 148);
123                 this.add(truckButton_4);
124         }
125
126     //gets the top 4 trucks by probability and segment
127     private Truck[] getTopTrucks(ResponsesDB responsesDB, Truck[] trucks){
128
129         //make a bayesian expert to ask
130         LogicExpert logicExpert = new LogicExpert();
131         //ask the expert to give the recommended trucks
132         Truck[] topTrucks = logicExpert.get4RecommendedBayesianAndSegmentTrucks(responsesDB, trucks)
    ;
133
134         //get the last truck and stuff it into the fourth place
135         Truck lastTruck = responsesDB.lastTruckPurchased;
136         if (lastTruck != null &&
137             lastTruck != topTrucks[0] &&
138             lastTruck != topTrucks[1] &&
139             lastTruck != topTrucks[2] &&
140             lastTruck != topTrucks[3]){
141             //replace the last truck
142             topTrucks[3] = responsesDB.lastTruckPurchased;
143             //and change the reason
144             topTrucks[3].recommendationReason = Truck.LAST_TRUCK_REASON;
145         }
146
147         return topTrucks;
148     }
149
150         //action method to deal with user hitting one of the truck image buttons
151         public boolean action(Event evt, Object arg){
152                 //dealing with a button being hit.
153                 if (button1.equals(evt.target) ){
154                     //leave the showroom
155                         showRoomPanel.remove(this);
156                         showRoomPanel.leaveShowRoom(ModuleState.DONE_VISIT);
157             return true;
158         }
159         else if (truckButton_1.equals(evt.target) ){
160                         showRoomPanel.remove(this);
161                         //show the first truck do a bayesian thing
162                         String[] reasons = topTrucks[0].getBayesianSalesFeatures();
163                         showRoomPanel.showTruckPanel(topTrucks[0]);
164                         return true;
165                 }
166                 else if (truckButton_2.equals(evt.target) ){
167                         showRoomPanel.remove(this);
168                         //do a bayesian thing here
169                         String[] reasons = topTrucks[2].getBayesianSalesFeatures();
170                         showRoomPanel.showTruckPanel(topTrucks[1]);
171                         return true;
```

```
172                 }
173                 else if (truckButton_3.equals(evt.target) ){
174                         showRoomPanel.remove(this);
175                         showRoomPanel.showTruckPanel(topTrucks[2]);
176                         return true;
177                 }
178                 else if (truckButton_4.equals(evt.target) ){
179
180                 showRoomPanel.remove(this);
181                         showRoomPanel.showTruckPanel(topTrucks[3]);
182                         return true;
183                 }
184                 return false;
185         }
186         //paint methods
187         //update or paint method here!!
188         //paint methods
189         public void update (Graphics g){
190                 paint (g);
191         }
192         public void paint (Graphics g){
193                 g.drawImage(this.backgroundImage, 0, 0, 800, 600, this);
194
195                 labHold.paintAll(g);
196         }
197
198 }
```

```
1 package Trucktown.ShowRoom;
2
3 import Trucktown.*;
4
5 import java.awt.*;
6
7 import symantec.itools.awt.LabelButton;
8 import symantec.itools.awt.WrappingLabel;
9
10 public class TruckPanel extends Panel
11 {
12         private ShowRoomPanel showRoomPanel;
13
14         symantec.itools.awt.WrappingLabel wrappingLabel1;
15         LtLabel label1;
16
17         /* these buttons are made public because
18            AutoMile reuses this panel and will need to
19            reorganize these buttons.
20
21         */
22         public ImageButton option;
23         public ImageButton meetPeople;
24         public ImageButton moneyButt;
25         public ImageButton testdrive;
26         public ImageButton comments;
27         public ImageButton compare;
28         public ImageButton mags;
29         public ImageButton specs;
30         public ImageButton mainMap;
31         public ImageButton toShowroomBut;
32         private boolean doneShow = false;
33         private Truck truck;
34         private Image truckImage;
35         private LabHolder labHold = new LabHolder();
36     Trucktown.Expert.FeaturesPanel featuresPanel = null;
37
38         //constructor
39         public TruckPanel(ShowRoomPanel showRoomPanel){
40                 this.showRoomPanel = showRoomPanel;
41                 setLayout(null);
42                 setSize(800,600);
43
44                 label1 = new LtLabel("Truck shown here",LtLabel.CENTER);
45                 label1.setBounds(130,5,540,60);
46                 label1.setFont(new Font("Dialog", Font.BOLD, 28));
47                 labHold.add(label1);
48
49                 //make "meet other people" image button
50                 Image meetOtherPeopleImage = showRoomPanel.parentFrame.util.getImage( "Images/Icons/
   Meet.gif" );
51                 meetPeople = new ImageButton(meetOtherPeopleImage);
52                 meetPeople.reshape(625, 150, 140, 70);
53                 this.add(meetPeople);
54
55                 //make "finantial info" image button
56                 Image financeButImage = showRoomPanel.parentFrame.util.getImage("Images/Icons/financ
   e.gif");
57                 moneyButt = new ImageButton(financeButImage);
58                 moneyButt.reshape(625, 230, 140, 70);
59                 this.add(moneyButt);
60
61                 //make "schedule test drive" image button
62                 Image scheduleTestDriveImage = showRoomPanel.parentFrame.util.getImage( "Images/Icon
   s/Schedule.gif" );
63                 testdrive = new ImageButton(scheduleTestDriveImage);
64                 testdrive.reshape(625, 310, 140, 70);
65                 this.add(testdrive);
66
67                 //make "see customer comments" image button
68                 Image customerCommentsImage = showRoomPanel.parentFrame.util.getImage( "Images/Icons
   /Comment.gif" );
69                 comments = new ImageButton(customerCommentsImage);
70                 comments.reshape(625, 390, 140, 70);
71                 this.add(comments);
72
73                 //make "take me to trucktown" image button
74                 Image backToTrucktownImage = showRoomPanel.parentFrame.util.getImage( "Images/Icons/
   Mainmap.gif" );
75                 mainMap = new ImageButton(backToTrucktownImage);
76                 mainMap.reshape(625, 470, 140, 70);
77                 this.add(mainMap);
78
79                 //make "optionconfig" image button
80                 Image optionConfigImage = showRoomPanel.parentFrame.util.getImage( "Images/Icons/Opt
   ion.gif" );
81                 option = new ImageButton(optionConfigImage);
82                 option.reshape(25, 150, 140, 70);
83                 this.add(option);
84
85                 //make "compare with other trucks" image button
86                 Image compareTrucksImage = showRoomPanel.parentFrame.util.getImage( "Images/Icons/Co
```

```
mpare.gif" );
                compare = new ImageButton(compareTrucksImage);
                compare.reshape(25, 230, 140, 70);
                this.add(compare);

                //make "magazine and newspaper articles" image button
                Image showArticlesImage = showRoomPanel.parentFrame.util.getImage( "Images/Icons/Mag
.gif" );
                mags = new ImageButton(showArticlesImage);
                mags.reshape(25, 310, 140, 70);
                this.add(mags);

                //make "specifications" image button
                Image specificationsImage = showRoomPanel.parentFrame.util.getImage( "Images/Icons/S
pecs.gif" );
                specs = new ImageButton(specificationsImage);
                specs.reshape(25, 390, 140, 70);
                this.add(specs);

                //make "take me to showroom" image button
                Image backToShowroomImage = showRoomPanel.parentFrame.util.getImage( "Images/Icons/S
howrm.gif" );
                toShowroomBut = new ImageButton(backToShowroomImage);
                toShowroomBut.reshape(25, 470, 140, 70);
                this.add(toShowroomBut);

        }

        //show it and do it!!
        public void show(Truck truck){
            this.truck = truck;
            //replace the title
            label1.setText(truck.GetString(truck.VEHICLE_STR));

            //remove this beast if it was here last
            if (featuresPanel != null) {
                this.remove(featuresPanel);
            }

        featuresPanel = new Trucktown.Expert.FeaturesPanel();
            featuresPanel.setBounds(233, 275, 334, 425);
            this.add(featuresPanel);
        if(truck.recommendationReason == Truck.BAYESIAN_REASON){
            featuresPanel.show(truck.getBayesianSalesFeatures(), null);
        }
        else if (truck.recommendationReason == Truck.SEGMENT_REASON){
            String[] tmp = new String[1];
            tmp[0] = truck.getSegmentSalesFeature();
            featuresPanel.show(tmp, truck.getBayesianSalesFeatures(), null);
        }
        else if (truck.recommendationReason == Truck.LAST_TRUCK_REASON){
            String[] tmp = new String[1];
            tmp[0] = Truck.lastTruckString;
            featuresPanel.show(tmp, null);
        }

        add(featuresPanel);

            //replace the image
        this.truckImage = showRoomPanel.parentFrame.util.getImage( truck.GetString(Truck.IMAGE_STR))
;
        //this.truckImage = showRoomPanel.parentFrame.util.getScaledImage(truckImage, 310, 202);


            doneShow = true;
            super.show();
        }

    public void reShow(){
        super.show();
    }

        //action method to deal with user hitting one of the truck image buttons
        public boolean action(Event evt, Object arg){
                //dealing with a button being hit.
                if (option.equals(evt.target) ){
                        showRoomPanel.remove(this);
                        showRoomPanel.showOptionConfigPanel(truck);
                        return true;
                }
                else if (specs.equals(evt.target) ){
                        //this will change
                        showRoomPanel.remove(this);
                        showRoomPanel.showSpecsPanel(truck);
                        return true;
                }
                else if (mags.equals(evt.target))
                {
                        showRoomPanel.remove(this);
                        showRoomPanel.showMagsPanel(truck);
                        return true;
                }
```

```
174                 else if (toShowroomBut.equals(evt.target) ){
175                         showRoomPanel.remove(this);
176                         showRoomPanel.showLastRoomPanel();
177                         return true;
178                 }
179                 else if (comments.equals(evt.target) ){
180                         showRoomPanel.remove(this);
181                         showRoomPanel.showCommentsPanel(truck);
182                         return true;
183                 }
184                 else if (compare.equals(evt.target) ){
185                         showRoomPanel.remove(this);
186             showRoomPanel.showComparePanel(truck);
187             return true;
188                 }
189                 else if (testdrive.equals(evt.target) ){
190
191                         showRoomPanel.remove(this);
192                         showRoomPanel.showScheduleTestPanel(truck);
193                         return true;
194                 }
195                 else if (moneyButt.equals(evt.target) )
196                 {
197                     showRoomPanel.remove(this);
198                         showRoomPanel.showFinancialPanel(truck);
199
200                         return true;
201                 }
202                 else if (meetPeople.equals(evt.target) ){
203
204                         showRoomPanel.remove(this);
205                         showRoomPanel.showMeetOthersPanel(truck);
206                         return true;
207                 }
208                 else if (mainMap.equals(evt.target) ){
209                         showRoomPanel.remove(this);
210                         showRoomPanel.leaveShowRoom(ModuleState.DONE_VISIT);
211                         return true;
212                 }
213                 return false;
214         }
215
216         //paint methods
217         //update or paint method here!!
218         public void update (Graphics g){
219                 paint (g);
220         }
221
222         public void paint (Graphics g){
223                 //paint the background
224                 g.drawImage(this.showRoomPanel.showRoomBackground, 0, 0, 800, 600, this);
225                 labHold.paintAll(g);
226
227                 if (doneShow)
228                 {
229                     g.setColor(Color.lightGray);
230                     //g.fill3DRect(200, 80, 385, 525, true);
231                     g.drawImage(this.truckImage, 272, 90, this);
232                 }
233         }
234 }
```

```
1  package Trucktown.Expert;
2
3  import Trucktown.*;
4
5  import java.awt.*;
6
7  public class WhatImThinkingPanel extends Panel{
8
9          protected ExpertPanel expertPanel;
10
11     //common expert objects.
12     //backgroung photo
13     protected Image whatImThinkingBackground;
14     protected Color buttonColor = new Color(16762880);
15     protected Font smallFont = new Font("Dialog", Font.BOLD, 14);
16     protected Font bigFont = new Font("Dialog", Font.BOLD, 20);
17
18     //main thinking panel
19     public WITPMainPanel witpMainPanel;
20
21         //list of trucks to be shown
22         protected Truck[] trucks;
23         //responsesDB to be used (at least for the segment room)
24         protected ResponsesDB responsesDB;
25
26         //this boolean is to be used everywhere
27         public boolean DEBUG = true;
28
29     Image backgroundImage;
30
31         //constructor
32         public WhatImThinkingPanel(ExpertPanel expertPanel){
33                 this.expertPanel = expertPanel;
34                 setLayout(null);
35                 setSize(800,600);
36
37         }
38
39         //show it and do it!!
40         //requires truck objects as inputs
41     //    gets truck array from SessionModel and calls showExpert()
42         public void showWhatImThinking(Image backgroundImage){
43             this.backgroundImage = backgroundImage;
44
45         //show the background while the person waits for images
46                 super.show();
47                 this.update(this.getGraphics());
48
49         //create the two main Panels - when implementing one showrooms
50         witpMainPanel = new WITPMainPanel(this);
51
52                 //show the intro panel
53                 showWITPMainPanel();
54                 super.show();
55         }
56
57         protected void showWITPMainPanel(){
58
59             this.add(witpMainPanel);
60             witpMainPanel.showWITPMainPanel(backgroundImage);
61         }
62
63         protected void showWITPTruckPanel(Truck truck){
64             this.remove(witpMainPanel);
65         //truck panel
66         WITPTruckPanel witpTruckPanel = new WITPTruckPanel(this);
67             this.add(witpTruckPanel);
68             witpTruckPanel.show(truck);
69         }
70
71         protected void finished(){
72                 expertPanel.remove(this);
73                 expertPanel.showQuestionsPanel();
74         }
75
76         //paint methods
77         //update or paint method here!!
78         public void update (Graphics g)
79         {
80                 paint (g);
81         }
82
83         public void paint (Graphics g)
84         {
85                 //paint the background
86         }
87 }
```

```
1 package Trucktown.Expert;
2
3 import Trucktown.*;
4 import java.awt.*;
5 import java.net.*;
6 import java.applet.*;
7 import symantec.itools.awt.WrappingLabel;
8
9 public class WITPMainPanel extends Panel{
10
11         private WhatImThinkingPanel whatImThinkingPanel;
12         private ExpertPanel expertPanel;
13         private Trucktown.ImageButton gotoConversationButton;
14
15         LtLabel truckLabel1;
16         LtLabel truckLabel2;
17         LtLabel truckLabel3;
18         LtLabel truckLabel4;
19     LabHolder labHold;
20
21         //these buttons should be changed to images when the resize tool is working correctly
22         private Trucktown.ImageButton truckButton_1;
23         private Trucktown.ImageButton truckButton_2;
24         private Trucktown.ImageButton truckButton_3;
25         private Trucktown.ImageButton truckButton_4;
26
27         private Truck[] recommendedTrucks;
28
29     private Image backgroundImage;
30
31         //constructor
32         public WITPMainPanel(WhatImThinkingPanel whatImThinkingPanel){
33             this.whatImThinkingPanel = whatImThinkingPanel;
34                 this.expertPanel = whatImThinkingPanel.expertPanel;
35                 setLayout(null);
36                 setSize(800,600);
37
38                 labHold = new LabHolder();
39
40                 LtLabel title1 = new LtLabel("I think these are the best trucks for you.", LghtLabel
   .CENTER);
41                 labHold.add(title1);
42                 title1.setBounds(148,15,485,25);
43                 title1.setFont(expertPanel.bigFont);
44
45                 LtLabel title2 = new LtLabel("Click on a truck's image", LghtLabel.CENTER);
46                 labHold.add(title2);
47                 title2.setBounds(148,40,485,25);
48                 title2.setFont(expertPanel.bigFont);
49
50                 LtLabel title3 = new LtLabel("to see why I recommended that truck.", LghtLabel.CENTE
   R);
51                 labHold.add(title3);
52                 title3.setBounds(148,65,485,25);
53                 title3.setFont(expertPanel.bigFont);
54
55                 Image conversationImage = expertPanel.parentFrame.util.getImage( "Images/Icons/backT
   oConversation.gif" );
56                 gotoConversationButton = new ImageButton(conversationImage);
57                 gotoConversationButton.reshape(310, 500, 140, 70);
58                 this.add(gotoConversationButton);
59
60                 //figure out what four trucks to show, requiring at least one truck to be above a
61                 //certain threshhold (like .1 a priori prob!)
62                 this.recommendedTrucks = getRecommendedTrucks(expertPanel.responsesDB, expertPanel.t
   rucks);
63                 //see if the first truck was filled in (to see if the experts returned anything)
64                 try {
65                     String s = recommendedTrucks[0].GetString(Truck.VEHICLE_STR);
66                 }
67                 catch (NullPointerException e){
68                     //didn't get a truck so just say go away (this should never happen anyway)
69
70                 //stop here before putting the trucks up
71                 return;
72                 }
73
74                 //first truck
75                 truckLabel1 = new LtLabel(recommendedTrucks[0].GetString(Truck.VEHICLE_STR), LghtLab
   el.CENTER);
76                 labHold.add(truckLabel1);
77                 truckLabel1.setBounds(105,120,250,30);
78                 truckLabel1.setFont(expertPanel.smallFont);
79
80         Image truckImage_1 = expertPanel.parentFrame.util.getImage( recommendedTrucks[0].GetString(T
   ruck.IMAGE_STR) );
81                 truckImage_1 = expertPanel.parentFrame.util.getScaledImage(truckImage_1, 225, 148);
82                 truckButton_1 = new ImageButton(truckImage_1);
83                 truckButton_1.reshape(110, 150, 225, 148);
84                 this.add(truckButton_1);
85
86                 //second truck
```

```
                truckLabel2 = new LtLabel(recommendedTrucks[1].GetString(Truck.VEHICLE_STR), LghtLabel.CENTER);
                labHold.add(truckLabel2);
                truckLabel2.setBounds(435,120,250,30);
                truckLabel2.setFont(expertPanel.smallFont);

        Image truckImage_2 = expertPanel.parentFrame.util.getImage( recommendedTrucks[1].GetString(Truck.IMAGE_STR) );
                truckImage_2 = expertPanel.parentFrame.util.getScaledImage(truckImage_2, 225, 148);
                truckButton_2 = new ImageButton(truckImage_2);
                truckButton_2.reshape(440, 150, 225, 148);
                this.add(truckButton_2);

                //third truck
                truckLabel3 = new LtLabel(recommendedTrucks[2].GetString(Truck.VEHICLE_STR), LghtLabel.CENTER);
                labHold.add(truckLabel3);
                truckLabel3.setBounds(105,310,250,30);
                truckLabel3.setFont(expertPanel.smallFont);

        Image truckImage_3 = expertPanel.parentFrame.util.getImage( recommendedTrucks[2].GetString(Truck.IMAGE_STR) );
                truckImage_3 = expertPanel.parentFrame.util.getScaledImage(truckImage_3, 225, 148);
                truckButton_3 = new ImageButton(truckImage_3);
                truckButton_3.reshape(110, 340, 225, 148);
                this.add(truckButton_3);

                //fourth truck
                truckLabel4 = new LtLabel(recommendedTrucks[3].GetString(Truck.VEHICLE_STR), LghtLabel.CENTER);
                labHold.add(truckLabel4);
                truckLabel4.setBounds(435,310,250,30);
                truckLabel4.setFont(expertPanel.smallFont);

        Image truckImage_4 = expertPanel.parentFrame.util.getImage( recommendedTrucks[3].GetString(Truck.IMAGE_STR) );
                truckImage_4 = expertPanel.parentFrame.util.getScaledImage(truckImage_4, 225, 148);
                truckButton_4 = new ImageButton(truckImage_4);
                truckButton_4.reshape(440, 340, 225, 148);
                this.add(truckButton_4);
        }

        //show it and do it!!
        public void showWITPMainPanel(Image backgroundImage){
            this.backgroundImage = backgroundImage;
            super.show();
        }

    //gets the top 2 trucks by probability
    //gets 2 segment trucks (regardless if user is still in more than one segment)
    private Truck[] getRecommendedTrucks(ResponsesDB responsesDB, Truck[] trucks){

        Truck[] topTrucks = expertPanel.logicExpert.get4RecommendedBayesianAndSegmentTrucks(responsesDB, trucks);
        return topTrucks;
    }

        //action method to deal with user hitting one of the truck image buttons
        public boolean action(Event evt, Object arg){
                //dealing with a button being hit.
                if (gotoConversationButton.equals(evt.target) ){
                    //leave the what im thinking panel
                        whatImThinkingPanel.finished();
            return true;
        }
        //pass to the truck panel the correct truck
        else if (truckButton_1.equals(evt.target)){
            //hit the first button
            whatImThinkingPanel.showWITPTruckPanel(recommendedTrucks[0]);
            return true;
        }
        else if (truckButton_2.equals(evt.target)){
            //hit the first button
            whatImThinkingPanel.showWITPTruckPanel(recommendedTrucks[1]);
            return true;
        }
        else if (truckButton_3.equals(evt.target)){
            //hit the first button
            whatImThinkingPanel.showWITPTruckPanel(recommendedTrucks[2]);
            return true;
        }
        else if (truckButton_4.equals(evt.target)){
            //hit the first button
            whatImThinkingPanel.showWITPTruckPanel(recommendedTrucks[3]);
            return true;
        }

                return false;
        }

        //paint methods
        //update or paint method here!!
```

```
172         //paint methods
173         public void update (Graphics g){
174                 paint (g);
175         }
176
177         public void paint (Graphics g){
178             //it looked like ass
179                 g.drawImage(this.backgroundImage, 0, 0, 800, 600, this);
180                 labHold.paintAll(g);
181         }
182 }
```

```
1 package Trucktown.Expert;
2
3 import Trucktown.*;
4 import java.awt.*;
5
6 public class WITPTruckPanel extends Panel
7 {
8
9         private WhatImThinkingPanel whatImThinkingPanel;
10
11        LtLabel label1;
12        LabHolder labHold;
13        ImageButton mainPanelButton;
14        private boolean doneShow = false;
15        private Truck truck;
16        private Image truckImage;
17    FeaturesPanel fPanel;
18        //constructor
19        public WITPTruckPanel(WhatImThinkingPanel whatImThinkingPanel){
20                this.whatImThinkingPanel = whatImThinkingPanel;
21                setLayout(null);
22                setSize(800,600);
23
24        labHold = new LabHolder();
25                label1 = new LtLabel("",LghtLabel.CENTER);
26                label1.setBounds(130,40,540,60);
27                label1.setFont(new Font("Dialog", Font.BOLD, 28));
28                labHold.add(label1);
29
30
31        fPanel = new FeaturesPanel();
32            fPanel.setBounds(425, 125, 334, 425);
33            this.add(fPanel);
34
35
36
37                //make "take me to main room" image button
38                Image backToMainImage = whatImThinkingPanel.expertPanel.parentFrame.util.getImage( "
  Images/Icons/back.gif" );
39                mainPanelButton = new ImageButton(backToMainImage);
40                mainPanelButton.reshape(150, 425, 75, 75);
41                this.add(mainPanelButton);
42
43        }
44
45        //show it with default text
46        public void show(Truck truck)
47        {
48            this.truck = truck;
49            //replace the title
50            label1.setText(truck.GetString(truck.VEHICLE_STR));
51            //replace the dialog
52            //fPanel.setBackground(java.awt.Color.lightGray);
53            if(truck.recommendationReason == Truck.BAYESIAN_REASON) {
54                fPanel.show(truck.getBayesianSalesFeatures(), truck.getBayesianSalesProblems());
55            }
56            else {
57                String[] tmp = new String[1];
58                tmp[0] = truck.getSegmentSalesFeature();
59                fPanel.show(tmp, truck.getBayesianSalesFeatures(), truck.getBayesianSalesProblems())
  ;
60            }
61
62            //replace the image
63        this.truckImage = whatImThinkingPanel.expertPanel.parentFrame.util.getImage( truck.GetString
  (Truck.IMAGE_STR));
64        this.truckImage = whatImThinkingPanel.expertPanel.parentFrame.util.getScaledImage(truckImage
  , 256, 167);
65
66            doneShow = true;
67            super.show();
68        }
69
70    public void reShow(){
71        super.show();
72    }
73
74        //action method to deal with user hitting one of the truck image buttons
75        public boolean action(Event evt, Object arg){
76                //dealing with a button being hit.
77                //this.remove(fPanel);
78                if (mainPanelButton.equals(evt.target) ){
79                        whatImThinkingPanel.showWITPMainPanel();
80                        whatImThinkingPanel.remove(this);
81                        return true;
82                }
83                return false;
84        }
85
86        //paint methods
87        //update or paint method here!!
88        public void update (Graphics g){
```

```
 89                 paint (g);
 90         }
 91
 92         public void paint (Graphics g){
 93                 //paint the background
 94                 g.drawImage(this.whatImThinkingPanel.backgroundImage, 0, 0, 800, 600, this);
 95                 if (doneShow){
 96                     g.setColor(Color.lightGray);
 97                     g.fill3DRect(25, 125, 345, 260, true);
 98                     g.drawImage(this.truckImage, 75, 172, this);
 99                     labHold.paintAll(g);
100                 }
101         }
102 }
```

```
1 package Trucktown.Expert;
2
3 import java.awt.*;
4 import symantec.itools.awt.WrappingLabel;
5 public class FeaturesPanel extends java.awt.Panel
6 {
7         TextArea reason1;
8         symantec.itools.awt.WrappingLabel titleLabel1;
9         TextArea reason2;
10        TextArea reason3;
11        symantec.itools.awt.WrappingLabel titleLabel2;
12        TextArea antireason1;
13        public static final Font reasonFont = new Font("Dialog", Font.PLAIN, 13);
14        public static final Font titleFont = new Font("Dialog", Font.BOLD, 18);
15
16        public FeaturesPanel()
17        {
18                setLayout(null);
19                setSize(334,425);
20                setBackground(java.awt.Color.lightGray);
21
22        }
23
24        public void show(String[] segmentF, String[] baysianF, String[] salesProblems) {
25            //this is invoked when the this is a segment truck
26        int reasonsLength = 0;
27        if (segmentF != null){
28            reasonsLength = segmentF.length;
29        }
30        if (baysianF != null){
31            reasonsLength = reasonsLength + baysianF.length;
32        }
33            //tack on the segment to the front of the baysian reasons
34            String[] reasons = new String[(segmentF.length + baysianF.length)];
35            int segmentLength = segmentF.length;
36            for (int i=0; i < segmentLength ;i++){
37                reasons[i] = segmentF[i];
38            }
39
40            for (int i=0 ; i < baysianF.length; i++){
41                reasons[i+segmentLength] = baysianF[i];
42            }
43            //and then send it to that constructor
44            show (reasons, salesProblems);
45        }
46
47
48        public void show(String[] features, String[] salesProblems) {
49
50        //deal with the sales problems
51        if((salesProblems != null) && (salesProblems.length > 0) && (salesProblems[0] != null) ) {
52            //make the label
53                titleLabel2 = new symantec.itools.awt.WrappingLabel();
54            try {
55                titleLabel2.setText("This truck is worth considering, even though this criterion is
  not satisfied:");
56                        titleLabel2.setFont(titleFont);
57            }
58            catch(java.beans.PropertyVetoException e) {}
59                titleLabel2.setBackground(Color.lightGray);
60                    titleLabel2.setBounds(24,276,288,72);
61                add(titleLabel2);
62            //add the sales problem
63                antireason1 = new TextArea(("*+salesProblems[0]), 4, 56, TextArea.SCROLLBARS_NONE);
64                antireason1.setBounds(24,348,288,72);
65                antireason1.setFont(reasonFont);
66                antireason1.setEditable(false);
67            antireason1.setBackground(Color.gray);
68                add(antireason1);
69        }
70
71            //send it to the next show
72            show(features);
73
74        }
75
76        public void show(String[] features) {
77
78        if( (features != null) && (features.length > 0) && (features[0] != null) ) {
79            //make the features label
80            titleLabel1 = new symantec.itools.awt.WrappingLabel();
81                try {
82                        titleLabel1.setText("Here are top reasons why I am recommending this truck:"
  );
83                }
84                catch(java.beans.PropertyVetoException e) { }
85                titleLabel1.setBackground(Color.lightGray);
86                titleLabel1.setBounds(24,12,288,48);
87                titleLabel1.setFont(titleFont);
88                add(titleLabel1);
89
90            //make the first reason
```

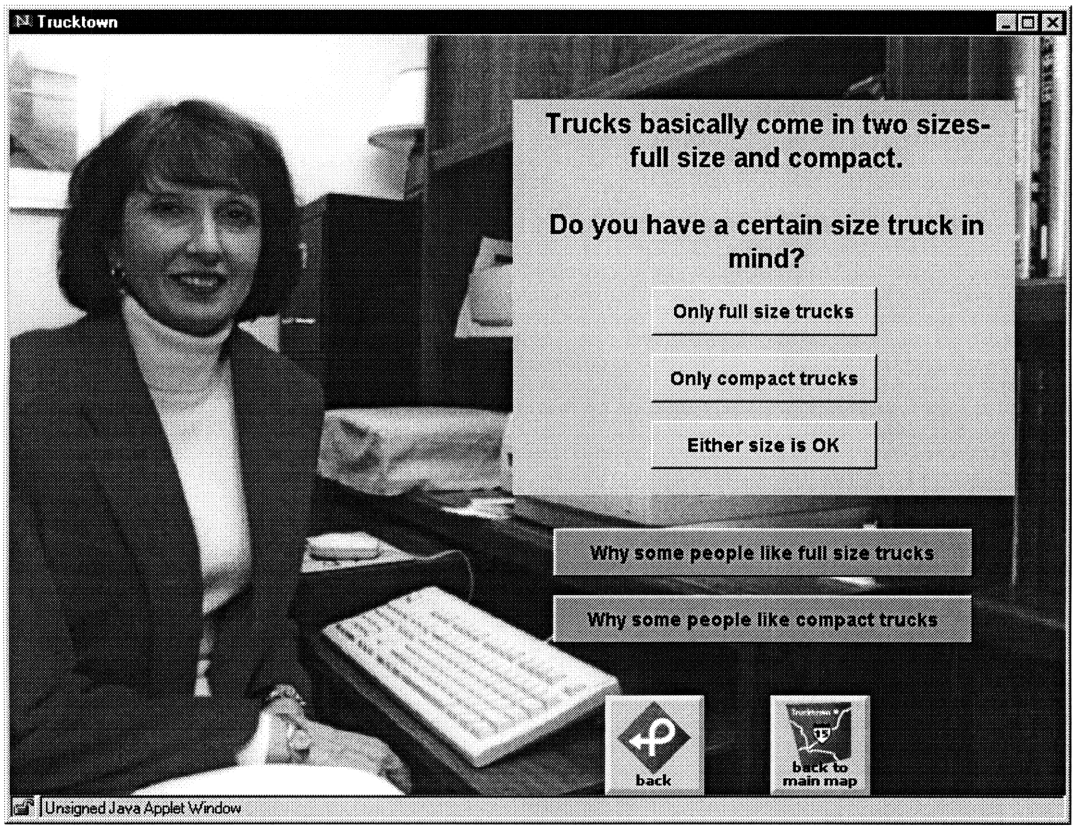
```
81                      expertPanel.showNeedMoreAnswers();
82                      return true;
83                  }
84
85                  return false;
86          }
87
88          //paint methods
89          //update or paint method here!!
90          //paint methods
91          public void update (Graphics g){
92                  paint (g);
93          }
94
95          public void paint (Graphics g){
96                  g.drawImage(this.expertPanel.expertBackground, 0, 0, 800, 600, this);
97          }
98  }
```

```
90              //make a bayesian expert to ask
91              BayesianExpert bayesianExpert = new BayesianExpert();
92                  //rank the questions and how they effected the response
93              /*
94              String selectedTruck = truckChoice.getSelectedItem();
95
96              //do in context
97              //get the index of the truck being investigated in the truck array
98              for (int truckIndex = 0; truckIndex < trucks.length; truckIndex++){
99                  if (selectedTruck == trucks[truckIndex].GetString(Truck.VEHICLE_STR)){
100                     //found the index to pass on
101                     //get the reasons for that truck
102                     ///!!!!!!!!!!AAAAAAAAAAAAAAAAAHHHHHHHHHHHHHHHHHHHHHHH IIIIIIIIIIII THOUGHT THIS
    WAS FIXXXXXXXXXXED!!!!
103
104                     this.responsesDB = bayesianExpert.makeDeltasWithContext(this.responsesDB, this.t
    rucks, truckIndex);
105                     this.responsesDB = bayesianExpert.rankResponsesBySalesDelta(this.responsesDB);
106                 }
107             }
108
109                 String truckString = "Delta's from truck: ";
110                 truckString = truckString + truckChoice.getSelectedItem();
111             g.drawString(truckString, 50, 50);
112             //for each response that has a delta
113             int totalShown = 0;
114             String responseString;
115             Response responseObj;
116             double formatedDelta;
117
118             for (int responseIndex = 1; responseIndex < responsesDB.responseArray.length ; responseI
    ndex++){
119                   //print it if it's got a value
120                   if (responsesDB.responseArray[responseIndex].delta != Response.NULL){
121                       responseObj = responsesDB.responseArray[responseIndex];
122                       //has a delta
123                       //quest:<#>  resp:<#>  rank<#>  delta:<#>
124                       responseString = "quest:";
125                       responseString = responseString + String.valueOf(responseObj.questionIndex);
126                       responseString = responseString + " resp:";
127                       responseString = responseString + String.valueOf(responseObj.response);
128                       responseString = responseString + " rank:";
129                       responseString = responseString + String.valueOf(responseObj.rank);
130                       responseString = responseString + " delta:";
131                       formatedDelta = (Math.rint(responseObj.delta*10000))/100;
132                       responseString = responseString + String.valueOf(formatedDelta) +"%";
133                       g.drawString(responseString, 20, (70 + 14*totalShown) );
134                       totalShown++;
135                   }
136             }
137
138             //do responses out of context
139             //get the index of the truck being investigated in the truck array
140             for (int truckIndex = 0; truckIndex < trucks.length; truckIndex++){
141                 if (selectedTruck == trucks[truckIndex].GetString(Truck.VEHICLE_STR)){
142                     //found the index to pass on
143                     //get the reasons for that truck
144                     ///!!!!!!!!!!AAAAAAAAAAAAAAAAAHHHHHHHHHHHHHHHHHHHHHHH IIIIIIIIIIII THOUGHT THIS
    WAS FIXXXXXXXXXXED!!!!
145
146                     this.responsesDB = bayesianExpert.makeDeltasWithoutContext(this.responsesDB, thi
    s.trucks, truckIndex);
147                     this.responsesDB = bayesianExpert.rankResponsesBySalesDelta(this.responsesDB);
148                 }
149             }
150
151             //for each response that has a delta
152             totalShown = 0;
153             for (int responseIndex = 1; responseIndex < responsesDB.responseArray.length ; responseI
    ndex++){
154                   //print it if it's got a value
155                   if (responsesDB.responseArray[responseIndex].delta != Response.NULL){
156                       responseObj = responsesDB.responseArray[responseIndex];
157                       //has a delta
158                       //quest:<#>  resp:<#>  rank<#>  delta:<#>
159                       responseString = "quest:";
160                       responseString = responseString + String.valueOf(responseObj.questionIndex);
161                       responseString = responseString + " resp:";
162                       responseString = responseString + String.valueOf(responseObj.response);
163                       responseString = responseString + " rank:";
164                       responseString = responseString + String.valueOf(responseObj.rank);
165                       responseString = responseString + " delta:";
166                       formatedDelta = (Math.rint(responseObj.delta*10000))/100;
167                       responseString = responseString + String.valueOf(formatedDelta) +"%";
168                       g.drawString(responseString, 300, (70 + 14*totalShown) );
169                       totalShown++;
170                   }
171             }
172                 */
173
174                 //going through each truck, prints the priors
175                 int i;
```

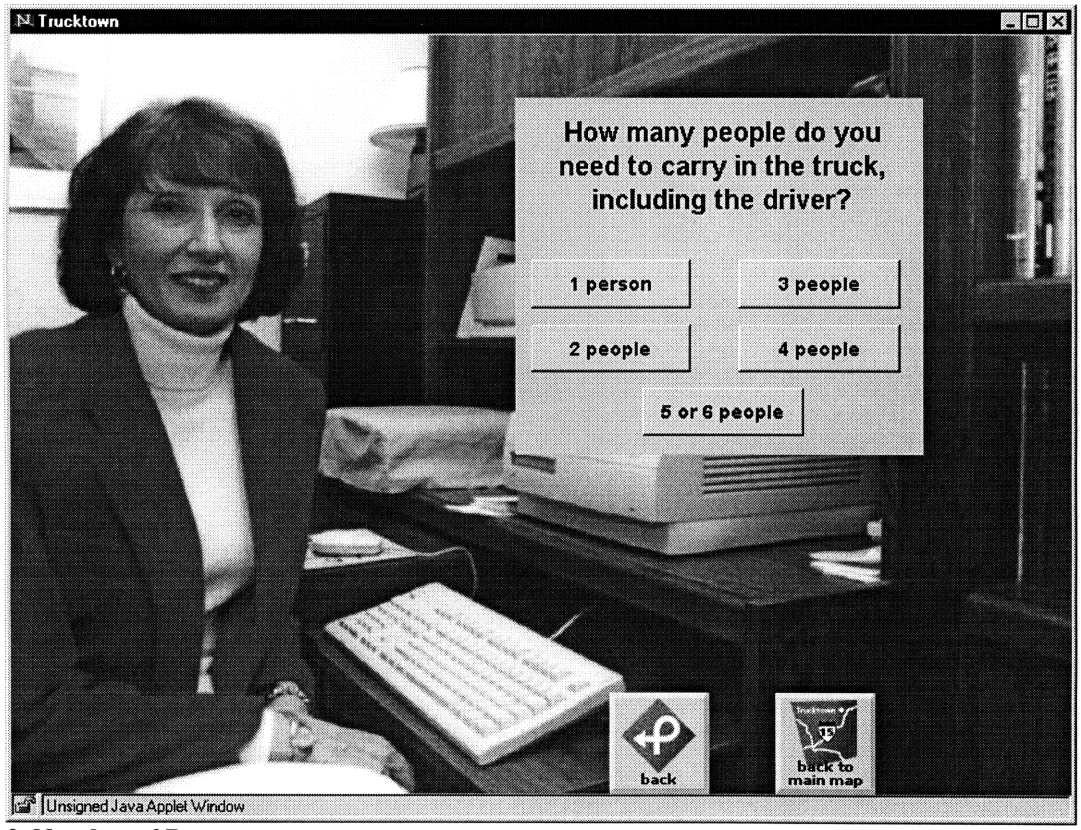
```
176                 String prioriProbString;
177                 double formatedPrioriProb;
178                 for (i = 0;i < (trucks.length/2); i++){
179                         g.drawString(trucks[i].GetString(Truck.VEHICLE_STR)+": ", 20, (60 +13*i));

180                         //show zero if it's reeally small
181                 formatedPrioriProb = (Math.rint(trucks[i].prioriProb*10000))/100;
182                     prioriProbString = Double.toString(formatedPrioriProb) +"%";

183                         g.drawString(prioriProbString, 200, (60 +13*i));
184
185                 }
186                 for (int secondHalf = i;secondHalf < (trucks.length); secondHalf++){
187                         g.drawString(trucks[secondHalf].GetString(Truck.VEHICLE_STR)+": ", 320, (60
    +13*(secondHalf-i)));
188                         //show zero if it's reeally small
189                 formatedPrioriProb = (Math.rint(trucks[secondHalf].prioriProb*10000))/100;
190                     prioriProbString = Double.toString(formatedPrioriProb) +"%";

191                         g.drawString(prioriProbString, 550, (60 +13*(secondHalf-i)));
192                 }
193         }//end of try
194         catch(Exception e){};
195         }
196 }
```

C. Expert Questions Screenshots

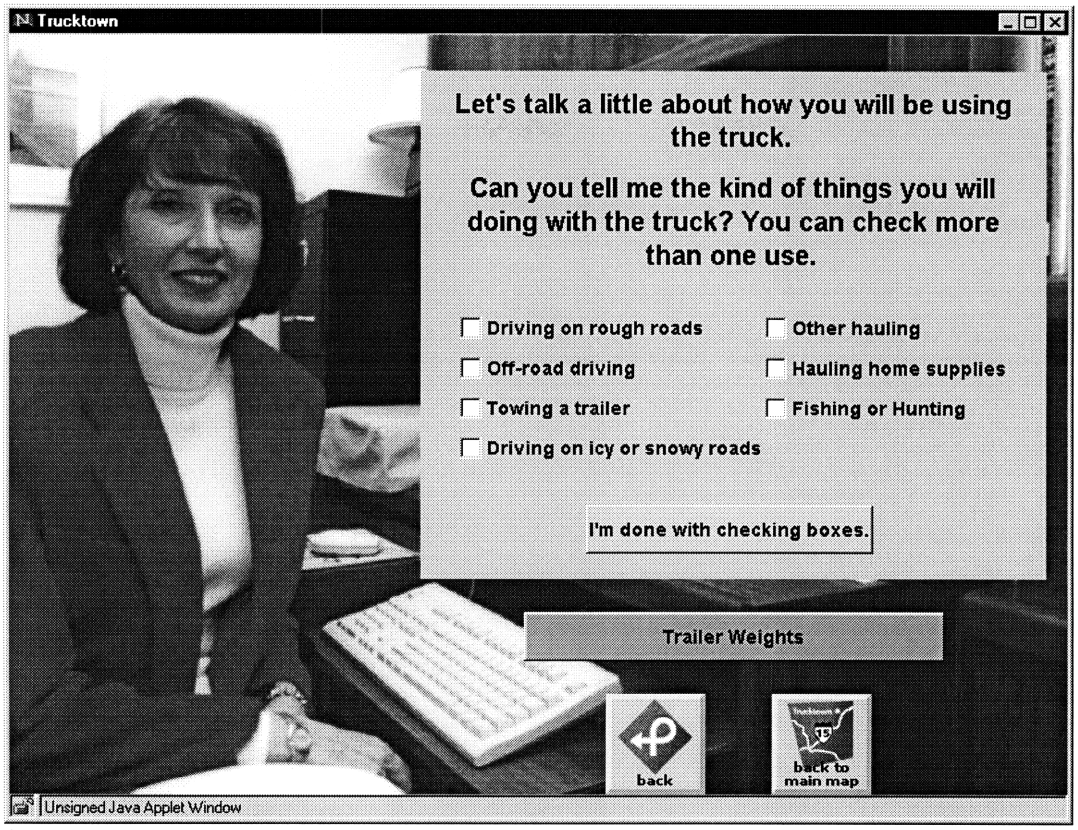
1. Size-Do you have a certain size truck in mind?
2. Number of Passengers-How many people do you need to carry in the truck?
3. Usage 1-Can you tell me the kind of things you will doing with the truck?
4. Usage Screen 2-Will you use the truck for any of these commercial uses?
5. Price-Did you have a budget in mind?
6. Manufacturers-Are there any truck manufacturers that you prefer?
7. Chip allocation-How do you feel about different truck qualities?
8. Passengers in front seat-How many people do you need to carry in the front seat of the truck?
9. Biggest Available-Do you prefer the biggest available or the smallest available vehicle?
10. Height-How tall is the tallest person who will ride in the truck?
11. Bed Length-Do you have a specific bed size in mind?
12. Number of Cylinders-What engine size or sizes do you prefer?
13. Comfortable Vehicle-How important is it that you have a big, quiet, comfortable vehicle?
14. Exterior look-Do you prefer a truck with a traditional or futuristic looking exterior?
15. Configure Options-What options would you like on your new truck?
16. Transmission-Did you have a specific transmission in mind?



1. Size



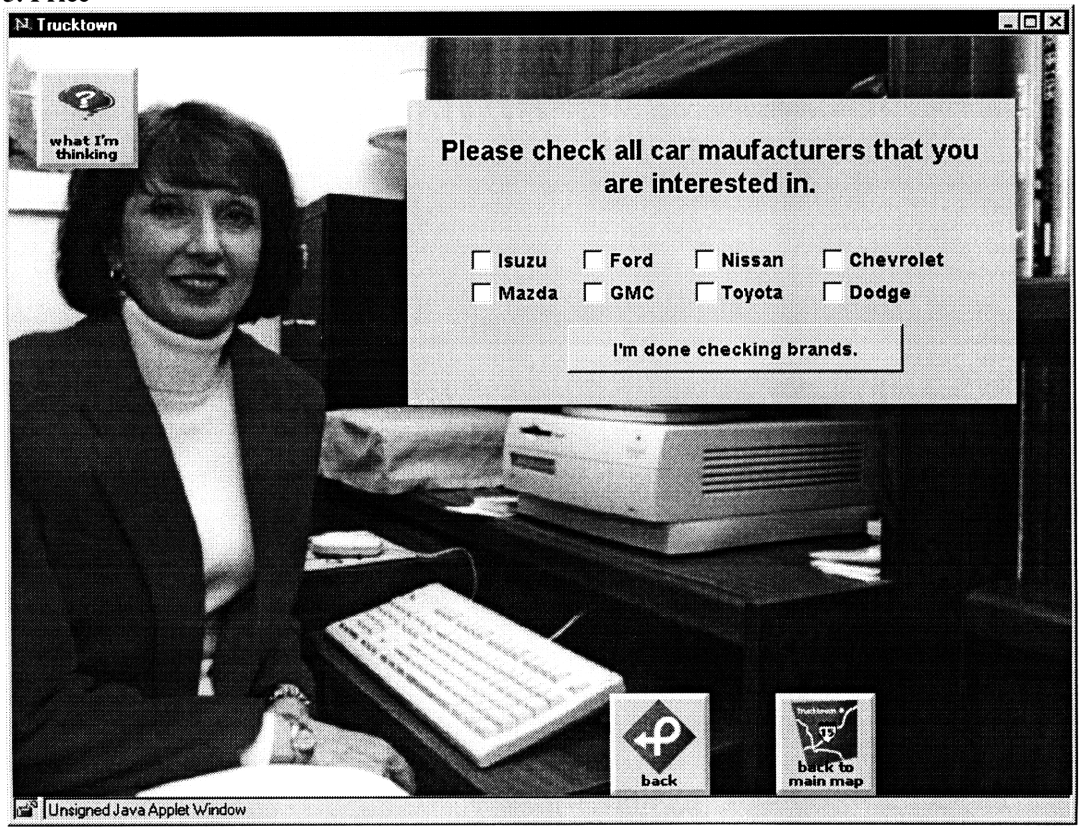
2. Number of Passengers



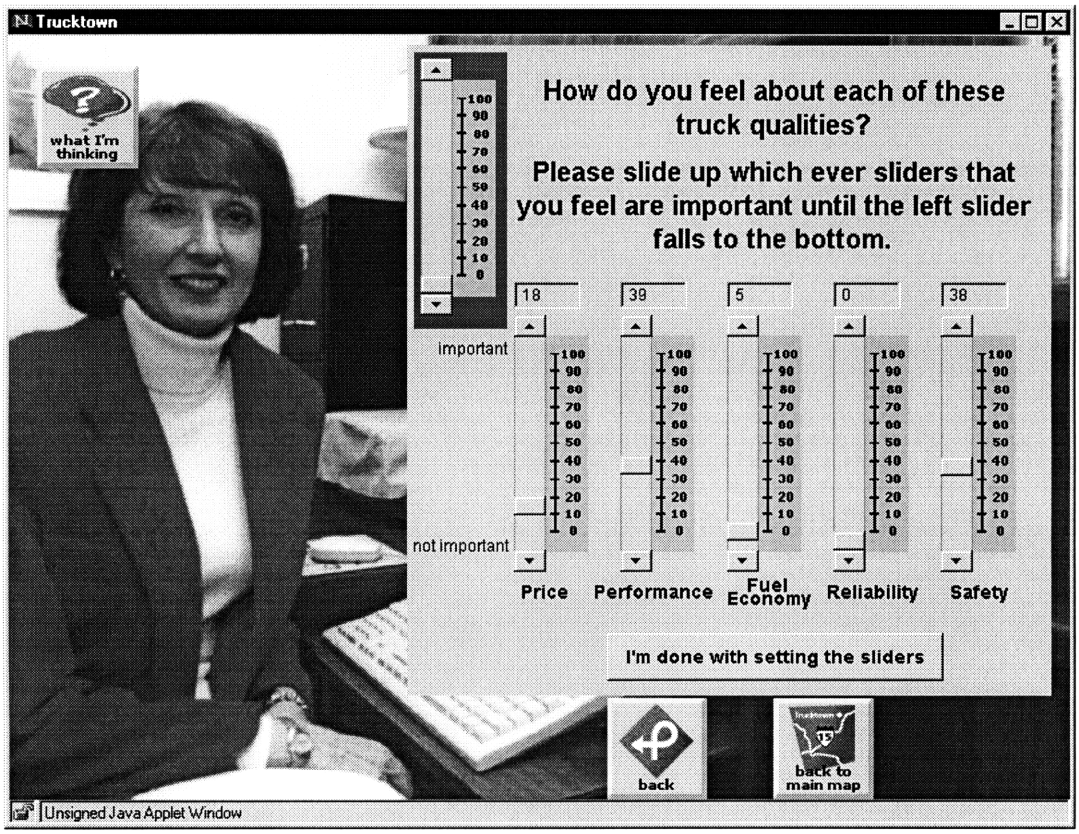
3. Usage 1

4. Commercial Usage 2

5. Price

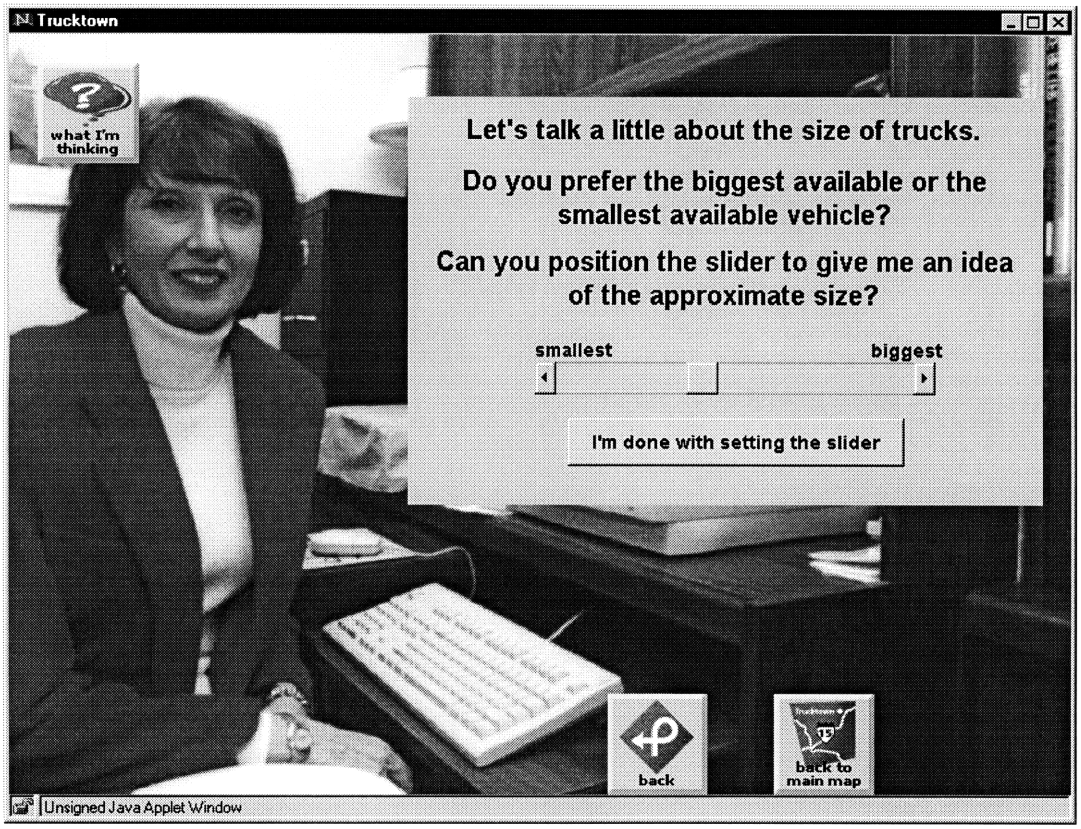


6.Manufacturers

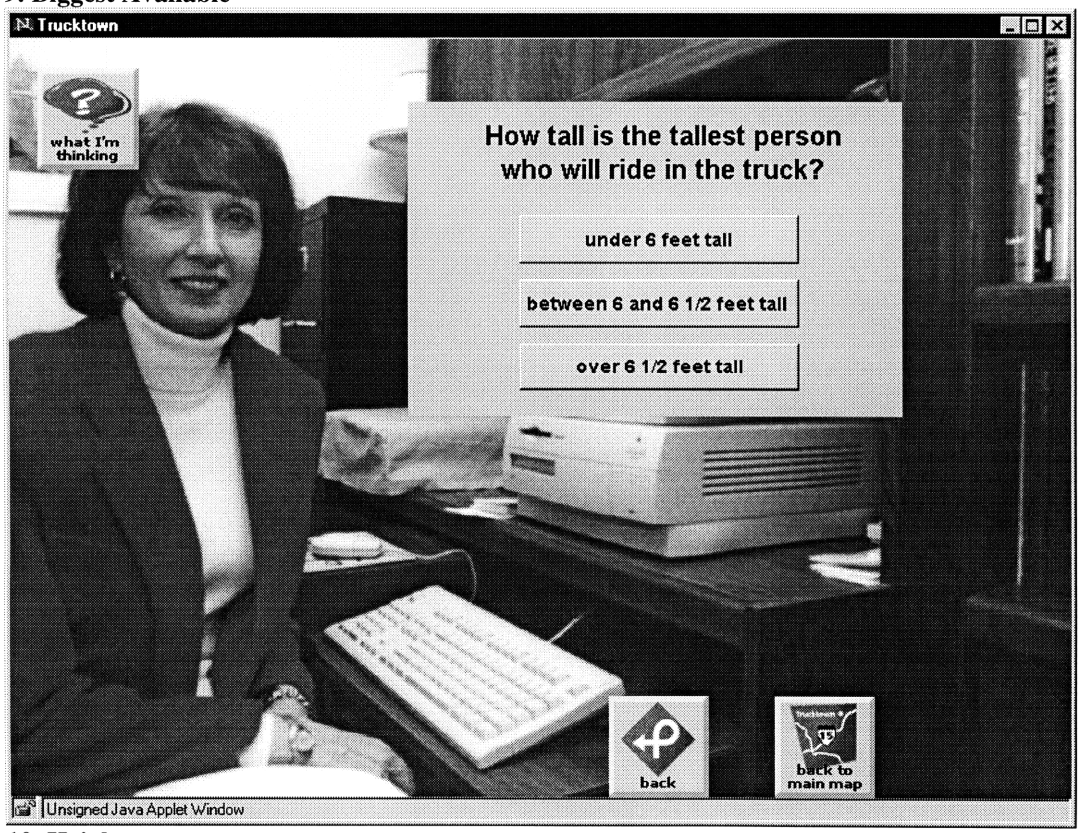


7. Chip Utility Allocation

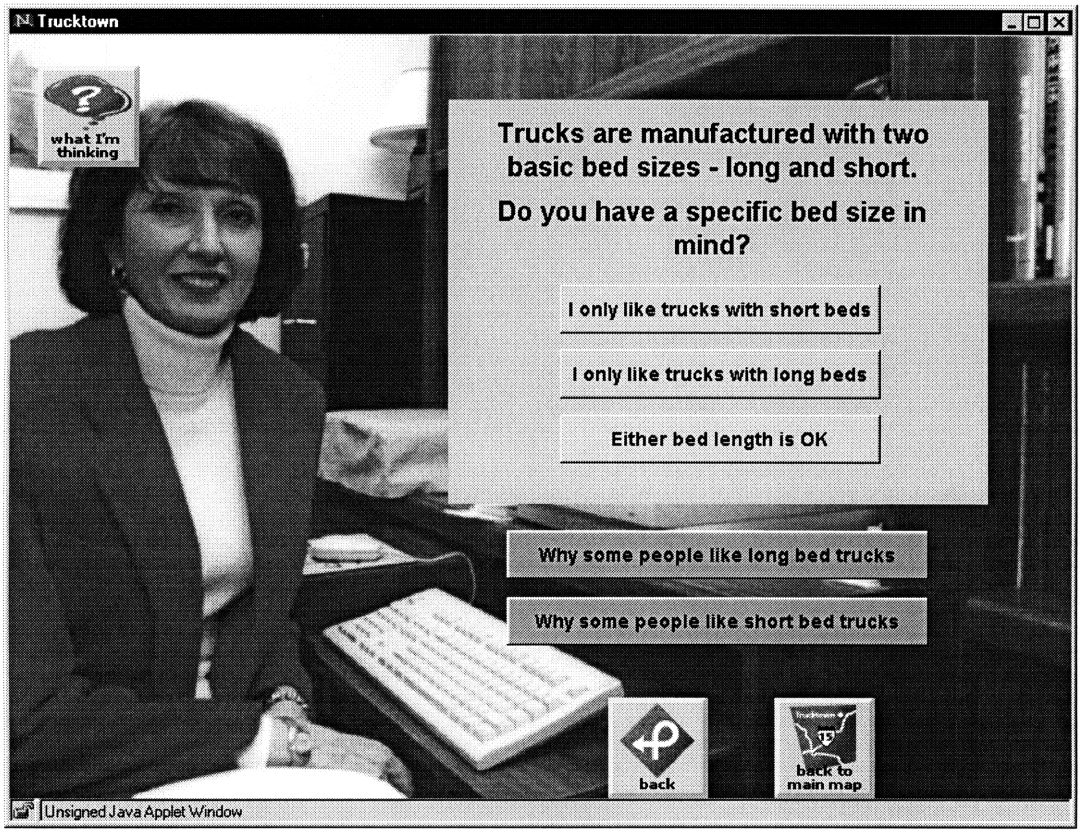
8. Passengers in Front Seat



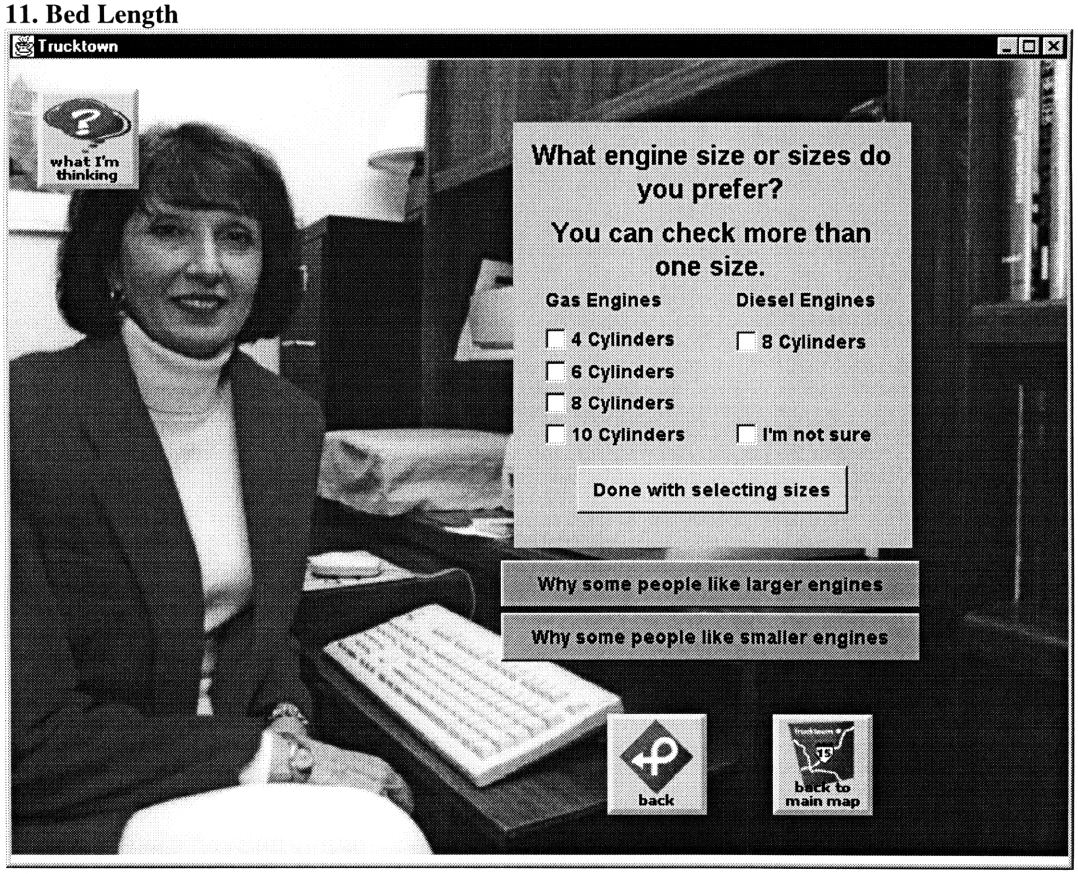
9. Biggest Available



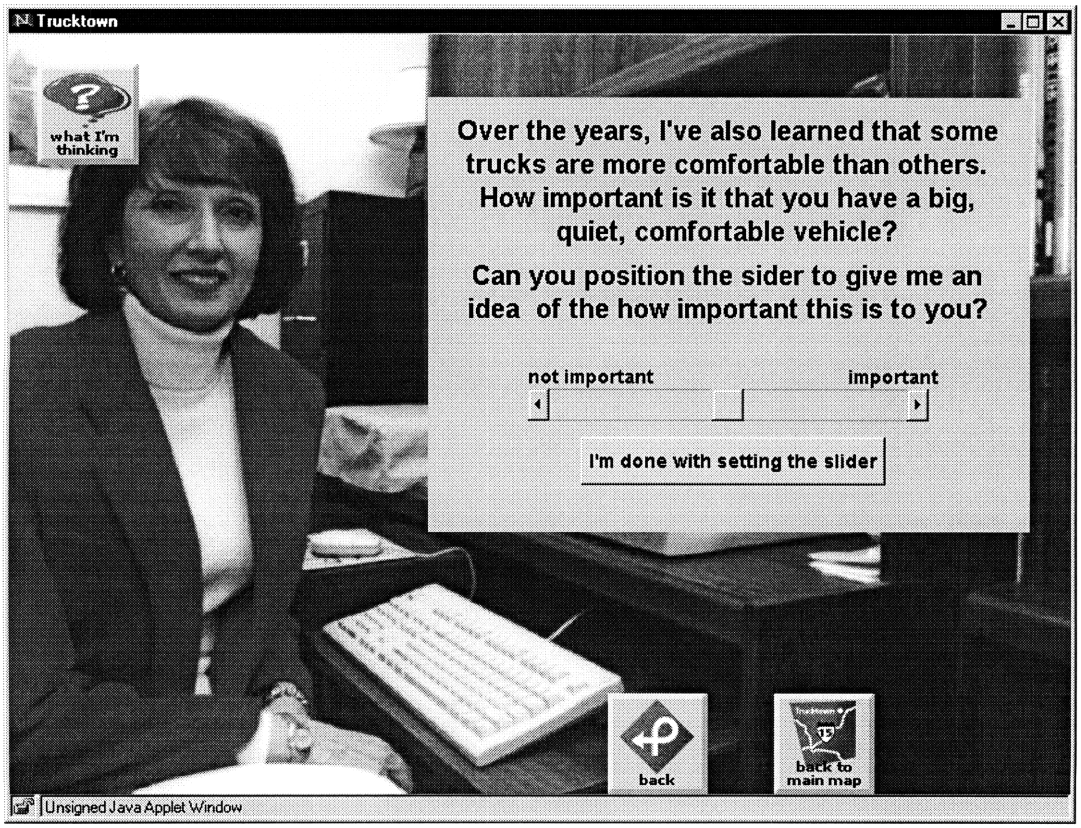
10. Height



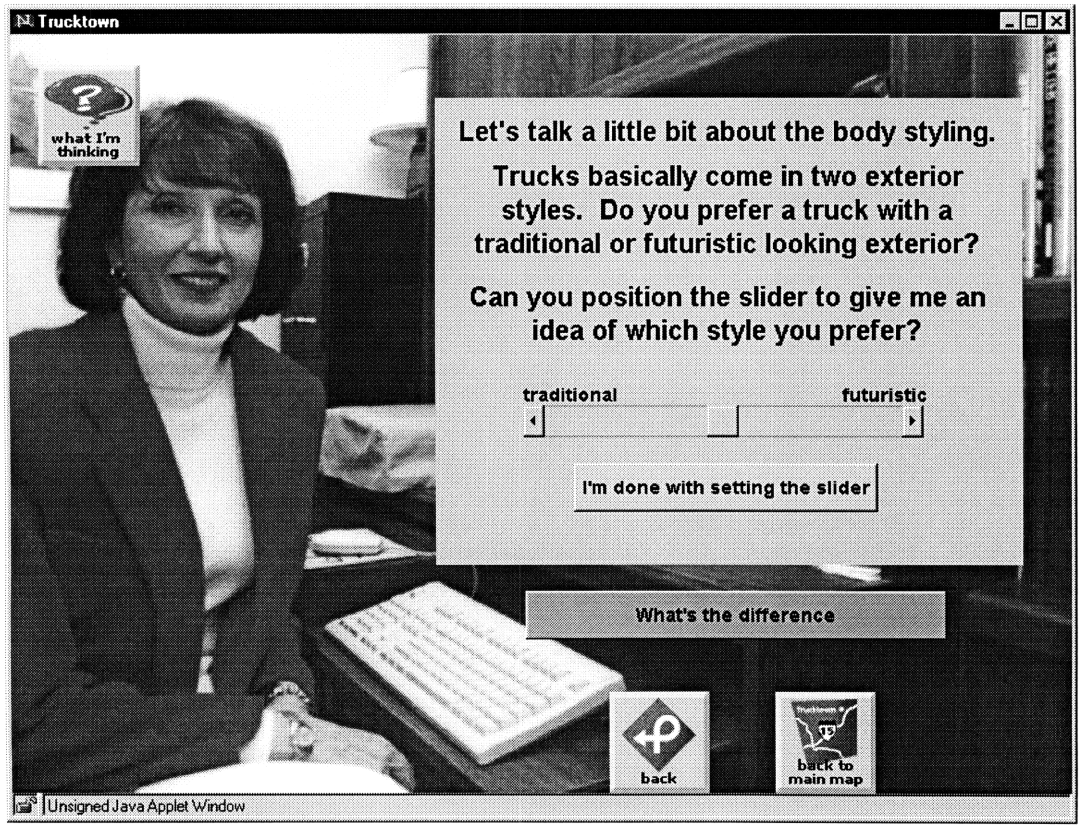
11. Bed Length



12. Number of Cylinders



13. Comfortable Vehicle



14. Exterior Look

D. General Trucktown, Server and Data Structure Code Listings

1. **TrucktownFrame.java**- High level container and controller of Trucktown.
2. **SessionModel.java**- Contains user session data and knowledge databases, including user responses and recommendation information.
3. **TruckDB.java**- Contains the truck objects.
4. **Truck.java**- Contains the truck data and recommendation information.
5. **Responses.java**- Contains all of the user responses after de-correlation.
6. **Response.java**- Contains user information and sales features data regarding this response.
7. **ExpertHistory.java**- Contains session history information.

```
1  package Trucktown;
2
3  import Trucktown.News.*;
4  import Trucktown.Expert.*;
5  import Trucktown.ShowRoom.*;
6  import Trucktown.AutoMile.*;
7  import java.awt.*;
8  import java.applet.*;
9  import java.awt.event.*;
10 import symantec.itools.awt.ImagePanel;
11 import symantec.itools.awt.ImageButton;
12 import java.util.Vector;
13 import java.util.Hashtable;
14 import ice.htmlbrowser.Browser;
15
16 public class TrucktownFrame extends Frame {
17
18     public String hostString = "http://vandelay.mit.edu/";
19     public String baseURL = "http://vandelay.mit.edu/Trucktown/";
20     public SessionModel data = null;
21     public SessionUtil util = null;
22
23     public boolean remote = true;
24     boolean debug = false;
25     boolean testGuide = true;
26     int top = 20;
27     Browser iceBrowser;
28
29     TrucktownApplet applet = null;
30     Button expertButton;
31     //Trucktown
32         //{{DECLARE_CONTROLS
33
34         Hashtable moduleTable;
35
36     // modules
37     MainMapPanel    mainMap;
38     ShowRoomPanel   showRoom;
39     ExpertPanel     mechanicExpert, neighborExpert, editorExpert;
40     TownHallPanel   townHall;
41     public CafePanel       cafe;
42     DinerPanel      diner;
43     //xing's stuff: 2/2/98
44     NewsPanel   newsstand;
45     AutoMilePanel   autoMile;
46     TestDrivePanel  testDrive;
47     MeetExpertsPanel meetExperts;
48     //}}
49     SpeechBubble speechBubble;
50
51     // reference to currently active module
52     Panel currentModule;
53 //   Button expertButton;
54
55  public TrucktownFrame(TrucktownApplet parentApplet) {
56     //added by Xing to add the password stuff.
57         /*      PasswordDialog pwdd = new PasswordDialog(this, "Enter User ID and Password");
58                 pwdd.show();
59                 */
60     //end of modification by Xing 2/17/98
61         iceBrowser = new Browser();
62
63         applet = parentApplet;
64         util = new SessionUtil(this);
65         data = new SessionModel(this);
66         moduleTable = new Hashtable();
67
68                 //{{INIT_CONTROLS
69                 setLayout(null);
70                 reshape(insets().left,insets().right,insets().left + insets().right + 800,insets().t
   op + insets().bottom + 600);
71                 setTitle("Trucktown");
72                 setResizable(true);
73
74                 speechBubble = new SpeechBubble(this, SpeechBubble.RIGHT);
75
76                 //the order that you add may end up being important!!
77                 moduleTable = new Hashtable();
78         mainMap = new MainMapPanel(this);
79                 add(mainMap);
80         mainMap.setVisible(true);
81                 moduleTable.put("MainMap",mainMap);
82
83         //      moduleTable.put("Guide",guide);
84
85     /* three experts, **Ken** needs to modify these panels!!!
86     */
87
88     // expert 1
89         mechanicExpert = new ExpertPanel(this, ExpertPanel.MECHANIC);
90         add(mechanicExpert);
91         mechanicExpert.setVisible(false);
```

```
92                  mechanicExpert.reshape(0, top, 800,600);
93                  moduleTable.put("MechanicExpert", mechanicExpert);
94
95          // expert 2
96          neighborExpert = new ExpertPanel(this, ExpertPanel.NEIGHBOR);
97          add(neighborExpert);
98          neighborExpert.setVisible(false);
99                  neighborExpert.reshape(0, top, 800,600);
100                 moduleTable.put("NeighborExpert", neighborExpert);
101
102         // expert 3
103         editorExpert = new ExpertPanel(this, ExpertPanel.EDITOR);
104         add(editorExpert);
105         editorExpert.setVisible(false);
106                 editorExpert.reshape(0, top, 800,600);
107                 moduleTable.put("EditorExpert", editorExpert);
108
109         //  end experts
110
111                 showRoom = new ShowRoomPanel(this);
112                 add(showRoom);
113                 showRoom.setVisible(false);
114                 showRoom.reshape(0, top, 800,600);
115                 moduleTable.put("ShowRoom",showRoom);
116
117             townHall = new TownHallPanel(this);
118                 add(townHall);
119                 townHall.setVisible(false);
120                 townHall.reshape(0, top, 800,600);
121                 moduleTable.put("TownHall", townHall);
122
123             cafe = new CafePanel(this);
124                 add(cafe);
125                 cafe.setVisible(false);
126                 cafe.reshape(0, top, 800,600);
127                 moduleTable.put("Cafe", cafe);
128
129             diner = new DinerPanel(this);
130                 add(diner);
131                 diner.setVisible(false);
132                 diner.reshape(0, top, 800,600);
133                 moduleTable.put("Diner", diner);
134
135             autoMile = new AutoMilePanel(this);
136                 add(autoMile);
137                 autoMile.setVisible(false);
138                 autoMile.reshape(0, top, 800,600);
139                 moduleTable.put("AutoMile", autoMile);
140
141             testDrive = new TestDrivePanel(this);
142                 add(testDrive);
143                 testDrive.setVisible(false);
144                 testDrive.reshape(0, top, 800,600);
145                 moduleTable.put("TestDrive", testDrive);
146
147                 meetExperts = new MeetExpertsPanel(this);
148                 add(meetExperts);
149                 meetExperts.setVisible(false);
150                 meetExperts.reshape(0, top, 800,600);
151             moduleTable.put("MeetExperts", meetExperts);
152
153             //xing's stuff: 2/2/98
154             newsstand = new NewsPanel(this);
155             add(newsstand);
156             newsstand.setVisible(false);
157             newsstand.reshape(0, top, 800, 600);
158             moduleTable.put("Newsstand", newsstand);
159
160                 // bottom most panel
161                 mainMap.setHotAreas();
162
163         data.initStateTable(moduleTable);
164
165         /*These window events will be passed to the processWindowEvent() method where
166           we can handle the user closing the window:
167         */
168         enableEvents(AWTEvent.WINDOW_EVENT_MASK);
169
170                 // on startup, load introduction
171         mainMap.guide.setDialogue("intro1");
172         mainMap.setEnabled(false);
173         moduleTable.put("Guide", mainMap.guide);
174         currentModule = mainMap;
175     }
176
177     protected void processWindowEvent(WindowEvent e) {
178         super.processWindowEvent (e);
179         if (e.getID () == WindowEvent.WINDOW_CLOSING)
180         setVisible (false);
181     }
182
183     public void hideModule(String moduleName) {
```

```
184             pln("hideModule: "+moduleName);
185             Component module = (Component) moduleTable.get(moduleName);
186             if (module != null) {
187             module.setVisible(false);
188             mainMap.setEnabled(true);
189             pln("done hiding module: "+moduleName);
190         }
191     }
192
193     public SpeechBubble getSpeechBubble() {
194
195         return speechBubble;
196     }
197
198     /*
199         Ask guide to check module states
200     */
201
202         public void goToModule(String moduleName) {
203
204             pln("goToModule: "+moduleName);
205
206             mainMap.setEnabled(false);
207             mainMap.setVisible(false);
208             // hides all other modules
209
210         /*    Component modules[] = getComponents();
211             for (int i = 0; i < modules.length; i++)
212                 modules[i].setVisible(false);
213           */
214             if (moduleName == "Exit") {
215                 // load up the exit dialog
216             mainMap.guide.setDialogue("Exit1");
217             mainMap.setVisible(true);
218             mainMap.setEnabled(false);
219             return;
220         }
221             speechBubble.setVisible(false);
222             speechBubble.dialogueView.setVisible(false);
223
224             if (currentModule instanceof ModulePanel) {
225             ((ModulePanel)currentModule).setVisible(false);
226         } else {
227             currentModule.setVisible(false);
228         }
229
230             Panel module = (Panel) moduleTable.get(moduleName);
231
232             /* 1/15: asks the guide to check currentModule's states, if the mainMap is the target
233                 module.
234                Guide just suggests destinations.
235
236             */
237
238         pln("asking the guide to check module state");
239             mainMap.guide.checkModuleState(data.getModuleName(currentModule), moduleName);
240
241             if (module instanceof ModulePanel) {
242                 ((ModulePanel)module).setDialogue(data.getWelcomeDialogue(moduleName));
243         }
244
245             if (mainMap.guide.isVisible()) {
246                 // if guide is visible, disable mainMap
247                 pln("guide is visible.");
248                 module.setVisible(false);
249             mainMap.setVisible(true);
250             mainMap.setEnabled(false);
251         } else if (module != null) {
252
253             // guide is NOT visible, and module isn't null
254
255             if (moduleName.equals("MainMap")) {
256
257                 // going to main map
258                 speechBubble.setVisible(false);
259                 speechBubble.dialogueView.setVisible(false);
260                 mainMap.setHelpButtonVisible(true);
261             } else {
262                 mainMap.setVisible(false);
263             }
264             // guide is NOT visible, just show the module
265                 module.setEnabled(true);
266                 module.setVisible(true);
267                 currentModule = module;
268
269                 pln("Done going to module: "+moduleName);
270                 if (debug) {
271                speechBubble.list();
272             }
273             }
274         }
275
```

```
276         /*
277             This is called by the experts and showroom
278         */
279
280         public void finishedWithModule(Panel module, int state) {
281
282             pln("finishedWithModule() called.");
283             data.setModuleState(module, state);
284         goToModule("MainMap");
285         }
286
287         public void shutDown() {
288
289             pln("Shutting down...");
290
291             Panel endSurveyPanel = new Panel();
292             endSurveyPanel.setLayout(null);
293             endSurveyPanel.setBounds(0,0,800,600);
294             endSurveyPanel.setBackground(Color.white);
295             Label helpLabel = new Label("Please raise your hand and ask for assistance.");
296             helpLabel.setFont(new Font("Dialog", Font.BOLD, 24));
297             helpLabel.setVisible(true);
298             helpLabel.setBounds(100,200,600,50);
299             endSurveyPanel.add(helpLabel);
300             add(endSurveyPanel,0);
301             endSurveyPanel.setVisible(true);
302         }
303
304     public void p(String s) {
305         if (debug)
306             System.out.print(s);
307     }
308
309     public void pln(String s) {
310         if (debug)
311             System.out.println("TrucktownFrame: "+s);
312     }
313
314     public void update(Graphics g) {
315
316         paint(g);
317     }
318
319     public TrucktownApplet getParentApplet(){
320         return applet;
321     }
322
323     public Browser getBrowserBean() {
324
325         if (iceBrowser == null) {
326             iceBrowser = new Browser();
327         }
328         iceBrowser.clearCache();
329                 return iceBrowser;
330
331     }
332 }
```

```
1  package Trucktown;
2
3  import Trucktown.Expert.*;
4  import java.awt.*;
5  import java.applet.*;
6  import java.util.*;
7
8  /*
9      This class encapsulates all the data used within a session.
10 */
11 public class SessionModel extends Observable{
12
13
14     boolean debug = false;
15     String dialogueFile = "Data/dialogue.txt";
16     String dialogueNamesFile = "Data/dialoguenames.txt";
17
18     TrucktownFrame parentFrame = null;
19
20     Truck[] chosenTrucks = null;
21
22     public TruckDB truckDB;
23
24     //added by kjgl 1-7-98
25     Truck[] segmentTrucks = null;
26     //Changed by kjgl 4-13-98
27     //the responseDb is the inner responsesDb (and can be reached by any of the topLayerDBs)
28     //it get's stomped by the topLayerResponseDB every time you go into an expert
29     public Trucktown.Expert.ResponsesDB responsesDB;
30     //3 response sets with the 3 experts
31     public Trucktown.Expert.TopLayerResponseDB editor_topLayerDB;
32     public ExpertHistory editor_expertHistory;
33
34     public Trucktown.Expert.TopLayerResponseDB neighbor_topLayerDB;
35     public ExpertHistory neighbor_expertHistory;
36
37     public Trucktown.Expert.TopLayerResponseDB mechanic_topLayerDB;
38     public ExpertHistory mechanic_expertHistory;
39
40     //says which expert was the last one who said anything
41     public int lastExpertRecommender;
42
43     // Added by irene 1/26/98
44     Vector testDriveInfo = new Vector();
45
46     // table of dialogue objects
47     Hashtable dialogueTable;
48     Hashtable welcomeDialogueTable;
49     // table keyed by each module
50     Hashtable stateTable;
51
52     Hashtable moduleState;
53     // table of all visit variables
54
55 //    Hashtable
56
57     // Loads up all the appropriate trucks
58     public SessionModel(TrucktownFrame frame) {
59
60         parentFrame = frame;
61
62         truckDB = new TruckDB(parentFrame.applet);
63         dialogueTable = new Hashtable();
64         loadDialogues();
65
66                 //make the responses DBs and expertHistories(not sure if the expert histories are a
   good idea)
67                 this.responsesDB = new Trucktown.Expert.ResponsesDB(parentFrame);
68         editor_topLayerDB = new Trucktown.Expert.TopLayerResponseDB(this.responsesDB);
69         editor_expertHistory = new Trucktown.Expert.ExpertHistory();
70         mechanic_topLayerDB = new Trucktown.Expert.TopLayerResponseDB(this.responsesDB);
71         mechanic_expertHistory = new Trucktown.Expert.ExpertHistory();
72         neighbor_topLayerDB = new Trucktown.Expert.TopLayerResponseDB(this.responsesDB);
73         neighbor_expertHistory = new Trucktown.Expert.ExpertHistory();
74
75         lastExpertRecommender = Trucktown.Expert.ExpertPanel.NONE;
76
77     }
78
79     public void initStateTable(Hashtable moduleTable) {
80
81         stateTable = new Hashtable();
82
83         Enumeration moduleNames = moduleTable.keys();
84         Enumeration modules = moduleTable.elements();
85         while (moduleNames.hasMoreElements()){
86             String name = (String) moduleNames.nextElement();
87             ModuleState newState = new ModuleState(name);
88             newState.moduleState = ModuleState.NO_VISIT;
89             stateTable.put(modules.nextElement(), newState);
90         }
91     }
```

```
92      public int getModuleState(String moduleName) {
93
94          Object state = stateTable.get(parentFrame.moduleTable.get(moduleName));
95          if (state == null)  {
96              pln("getModuleState: Module "+moduleName+" not found!!!");
97              pln(stateTable.toString());
98              return -1;
99          }
100         else
101             return ((ModuleState)state).moduleState;
102     }
103
104     public boolean setModuleState(Panel module, int i) {
105
106         ModuleState state = (ModuleState) stateTable.get(module);
107         if (state == null) {
108             return false;
109         }
110         else  {
111             state.moduleState = i;
112             return true;
113         }
114     }
115
116     public String getModuleName(Panel module) {
117         ModuleState state = (ModuleState) stateTable.get(module);
118
119         if (state == null) {
120             return "";
121         }
122         else  {
123             return state.moduleName;
124         }
125     }
126
127     public String getWelcomeDialogue(String moduleName) {
128
129         return moduleName+"1";
130     }
131
132     /*
133         Loads up the dialogue table from a Properties file:
134
135         File format:
136             identifier=content
137
138         identifier format:
139             intro1_T_1      dialogue text
140             intro1_L_0      response label
141             intro1_A_0      response action
142
143         where:
144             intro1  dialogue name
145             T       dialogue type
146             1       number of responses
147     */
148
149     public void loadDialogues() {
150
151         String currTag, dialogueNames;
152         String identifier, name, type, content;
153         StringTokenizer tokenizer,nameTokens;
154         int index, numResponses;
155         Dialogue newDialogue = null;
156         Enumeration enum;
157
158
159         /*
160             Creates all dialogue objects
161         */
162         // loads the dialoguenames
163         Properties namesProperties =  parentFrame.util.getPropertiesFile(dialogueNamesFile);
164
165         Enumeration allTags = namesProperties.keys();
166         while (allTags.hasMoreElements()) {
167             currTag = (String)allTags.nextElement();
168             dialogueNames = (String)namesProperties.remove(currTag);
169
170             pln("currTag = "+currTag+"\n dialogueNames = "+dialogueNames);
171             tokenizer = new StringTokenizer(dialogueNames, "-");
172             // creats all the dialogue objects
173             while (tokenizer.hasMoreTokens()) {
174
175                 identifier = tokenizer.nextToken();
176                 pln("identifier = "+identifier);
177                 nameTokens = new StringTokenizer(identifier, "_");
178                 name = nameTokens.nextToken();
179
180                 // what if the dialogue has 0 responses?
181                 numResponses = Integer.parseInt(nameTokens.nextToken());
182                 newDialogue = new Dialogue();
183
```

```
184                 dialogueTable.put(name, newDialogue);
185                 newDialogue.labels = new String[numResponses];
186                 newDialogue.actions = new String[numResponses];
187                 pln("Dialogue reated: "+name);
188             }
189         }
190         /*
191             Parses the actual dialogue objects
192         */
193         Properties dialogueProp = parentFrame.util.getPropertiesFile(dialogueFile);
194
195         enum = dialogueProp.propertyNames();
196         while(enum.hasMoreElements()) {
197             identifier = (String) enum.nextElement();
198             tokenizer = new StringTokenizer(identifier, "_");
199
200             name = tokenizer.nextToken();
201             type = "";
202             try {
203                 type = tokenizer.nextToken();
204             } catch (Exception e) {
205                 pln(e.toString());
206             }
207             pln("name = "+name);
208           index = Integer.parseInt(tokenizer.nextToken());
209             content = dialogueProp.getProperty(identifier);
210             pln("type = "+ type+" | index = "+index+
211                 " | content = "+content);
212             /*
213                ex:
214                intro1_T_1
215                there is 1 response
216                */
217             newDialogue = (Dialogue) dialogueTable.get(name);
218             if (type.equals(Dialogue.TEXT)) {
219     //             content = content.replace('$', (char)Character.LINE_SEPARATOR);
220                 newDialogue.text = content;
221             } else
222             /* ex:
223                 intro1_L_0
224                 the label at index 0
225              */
226             if (type.equals(Dialogue.LABEL)) {
227                try {
228                 newDialogue.labels[index] = content;
229                 } catch (Exception e) {
230                     pln("error in dialogue: "+content);
231                 }
232             } else
233             /* ex:
234                 intro_A_0
235                 the action at index 0
236              */
237             if (type.equals(Dialogue.ACTION)) {
238                 newDialogue.actions[index] = content;
239             }
240         }
241     }
242
243     /*
244         Truck methods
245     */
246     // get methods  -- start
247     public Truck getTruckFromName(String name) {
248                 return truckDB.getTruckFromName(name);
249         }
250
251         public Truck[] getAllTrucks() {
252             return truckDB.getAllTrucks();
253         }
254
255         public Truck getTruckFromIndex(int index)
256         {
257             return truckDB.GetTruck(index);
258         }
259
260     public Truck[] getChosenTrucks() {
261         return chosenTrucks;
262     }
263
264     public Dialogue getDialogue(String name) {
265
266         Object obj = dialogueTable.get(name);
267         if (obj == null) {
268             return null;
269         } else
270             return (Dialogue) obj;
271     }
272     // get methods  -- end
273
274
275     public void setChosenTrucks(Truck[] trucks) {
```

```
276
277         chosenTrucks = trucks;
278     }
279
280     // Added by Irene 1/25 ***************************************
281     public void addSchedule(Schedule sch)
282     {
283         testDriveInfo.addElement(sch);
284     }
285
286     public Vector getTestDrives()
287     {
288         return testDriveInfo;
289     }
290     // End of Irene's additions***********************************
291
292
293     //array of references to trucks in the user's segment
294     //added by kjgl 1-7-98
295     public void setSegmentTrucks(Truck[] segmentTrucks){
296         this.segmentTrucks = segmentTrucks;
297     }
298
299     public Truck[] getSegmentTrucks(){
300         return this.segmentTrucks;
301     }
302     //end segment stuff
303
304     public void p(String s) {
305
306         if (debug)
307             System.out.print(s);
308     }
309     public void pln(String s) {
310
311         if (debug)
312             System.out.println("SessionModel: "+s);
313     }
314
315 }
```

```
1  package Trucktown.Expert;
2
3  import Trucktown.*;
4
5  //added by Xing 2/27/98
6  import java.rmi.*;
7  import java.rmi.server.*;
8  import java.applet.Applet;
9  import Trucktown.Server.ResponseDBRM;
10 //
11 //
12 // ResponsesDB
13 //
14 //
15 public class ResponsesDB {
16         boolean debug = false;//added by xing 2/27/98
17
18         private TrucktownFrame parentFrame;
19         //the int responses start out life as -1 to make it easier to know if they have ever
20         //  been answered (would have used null but those are not primatives)
21         //they are in a response array (zero reserved to be nothing)
22         public Response[] responseArray;
23         private int numberOfQuestions = 41;
24         //deal with chip allocation answers in a speacial way(asuming that they
25     //are not going to be used in the bayesian update
26     //default them to be 20
27         public double response_MRSP = 20;
28         public double response_FuelEconomy = 20;
29         public double response_Dependability = 20;
30         public double response_Safety = 20;
31         public double response_HorsePower = 20;
32
33         //to be calculated once the chip responses are filled in.
34         public double normalizedResponse_MRSP;
35         public double normalizedResponse_FuelEconomy;
36         public double normalizedResponse_Dependability;
37         public double normalizedResponse_Safety;
38         public double normalizedResponse_HorsePower;
39
40         //where to put the segment array that the segment expert gives back
41         public int[] cachedSegArray = {1,2,3,4,5,6,7,8};
42
43     // Added by irene 2/16/98
44     public Truck lastTruckPurchased = null;
45
46     //values that give meaning to the response Database
47     public static final int NULL_VAL                    = 0;
48     public static final int FULLORCOMPACT_VAL           = 1;    // question panel 1 (3 vals)
49     public static final int PRICERANGE_VAL              = 2;    // question panel 5 (8 vals)
50     public static final int OFFROAD_VAL                 = 3;    // question panel 3 (2 vals ea.)
51     public static final int TOW_VAL                     = 4;    // question panel 3
52     public static final int HAULING_VAL                 = 5;    // question panel 3
53     public static final int CONSTRUCTION_VAL            = 6;   // question panel 4 (2 vals ea.)
54     public static final int SNOWPLOWING_VAL             = 7;   // question panel 4
55     public static final int NUMBERPEOPLE_VAL            = 8;   // question panel 2 (5 vals)
56     public static final int PEOPLEINFRONTSEAT_VAL       = 9;   // question panel 8 (3 vals)
57     public static final int TRUCKSIZE_VAL               = 10;   // question panel 9 (5 vals)
58     public static final int LIKESCHEVY_VAL              = 11;   //  question panel 6 (2 vals ea.)
59     public static final int LIKESDODGE_VAL              = 12;
60     public static final int LIKESFORD_VAL               = 13;
61     public static final int LIKESGMC_VAL                = 14;
62     public static final int LIKESISUZU_VAL              = 15;
63     public static final int LIKESMAZDA_VAL              = 16;
64     public static final int LIKESNISSAN_VAL             = 17;
65     public static final int LIKESTOYOTA_VAL             = 18;
66     public static final int BEDSIZE_VAL                 = 19;   // question panel 11 (2 vals)
67     public static final int TALLESTUSER_VAL             = 20;   // question panel 10 (3 vals)
68     public static final int FOURCYLGASENGINE_VAL        = 21;   // question panel 12 (2 vals ea.)
69     public static final int SIXCYLGASENGINE_VAL         = 22;   // question panel 12
70     public static final int EIGHTCYLGASENGINE_VAL       = 23;   // question panel 12
71     public static final int TENCYLGASENGINE_VAL         = 24;   // question panel 12
72     public static final int EIGHTCYLDIESEL_VAL          = 25;   // question panel 12
73     public static final int COMFORT_VAL                 = 26;   // question panel 13 (5 vals)
74     public static final int BODYSTYLING_VAL             = 27;   // question panel 14 (5 vals
75     public static final int GREATSTERIO_VAL             = 28;   //  options panels
76         public static final int THRDFRTHDOOR_VAL            = 29;
77         public static final int LEATHERSEATS_VAL            = 30;
78         public static final int CRUISECONTROL_VAL           = 31;
79         public static final int POWERLOCK_VAL               = 32;
80         public static final int POWERWINDOW_VAL             = 33;
81         public static final int TILTWHEEL_VAL               = 34;
82         public static final int TINTWINDOW_VAL              = 35;
83         public static final int THEFTDETER_VAL              = 36;
84         public static final int SAFETYUNIMP_VAL             = 37;
85         public static final int BETTERTIRES_VAL             = 38;
86         public static final int ANTILOCKBRAKES_VAL          = 39;
87         public static final int FOURWHEELDR_VAL             = 40;
88         public static final int AUTOMATIC_VAL               = 41;
89
90
91     /*
92         each response int in the response array needs to be replaced with a response
```

```
93          object.  this object will store a selling point as to why each
94          response was important. The
95
96          These objects will also be responsible for validating how many responses
97          are in the question and has a routine that validates it's response array in
98          the truck object as having correctly formated information.
99
100     */
101
102
103
104         //constructor
105         public ResponsesDB(TrucktownFrame parentFrame){
106             String url ="";
107                 this.parentFrame = parentFrame;
108                 //fill the response array enough to have 47 questions with 0 being null.
109             Applet theApp = this.parentFrame.getParentApplet();
110
111                 try {
112                     if (parentFrame.remote) {
113                     //added by xing.
114                     url = "rmi://vandelay.mit.edu/";
115                 } else {
116                     url = "rmi://"+(theApp.getDocumentBase()).getHost()+"/";
117                     }
118                     ResponseDBRM rsDB = (ResponseDBRM) Naming.lookup(url+"THE_RESPONSEDB");
119                     pln("the remote responseDB obtained");
120                     responseArray = rsDB.getResponses();
121                     pln("responseArray obtained");
122
123                 }
124                 catch(Exception re) {
125                     error("in constructor:"+re);
126                 }
127         }
128
129     public void cleanSalesInfo(){
130
131         //clean all of the responses of sales info
132         for(int i = 1; i < responseArray.length; i++){
133             //clean the insides
134             responseArray[i].delta = Response.NULL;
135             responseArray[i].salesDelta = Response.NULL;
136             responseArray[i].rank = -1;
137         }
138     }
139
140     //can be used to get values out of the response array
141     public int getVal(int index)
142     {
143         if((index >= 0) && (index < numberOfQuestions))
144             return responseArray[index].response;
145
146         else return 0;
147     }
148
149
150         public void print(){
151             for (int i = 1; i<responseArray.length; i++){
152                     System.out.println("response_:" +i+": "+ responseArray[i].response);
153                 }
154         }
155
156         public void validateResponseArray(){
157             //look through all the entries for strange entries
158             for (int i = 1; i<responseArray.length; i++){
159                 if (responseArray[i].response < 1){
160                     //response was not filled in
161                     if (responseArray[i].response != -1){
162                         //response was not a legitimate entry
163                         error("there was a problem at " + i);
164                     }
165                 }
166                 }
167         }
168
169         //added by irene 2.9.98
170     // produces a dummy ResponsesDB for testing purposes
171     public ResponsesDB getDummyResponsesDB()
172     {
173         ResponsesDB dummyDB = new ResponsesDB(parentFrame);
174
175         // don't care.
176         dummyDB.responseArray[FULLORCOMPACT_VAL].response           = 3;    // question panel 1 (3 v
    als)
177         // (1-8); 1 low, 8 high
178         dummyDB.responseArray[PRICERANGE_VAL].response              = 2;    // question panel 5 (8 v
    als)
179         dummyDB.responseArray[OFFROAD_VAL].response               = 2;    // question panel 3 (2 val
    s ea.)
180         dummyDB.responseArray[TOW_VAL].response              = 2;    // question panel 3
181         dummyDB.responseArray[HAULING_VAL].response           = 2;    // question panel 3
```

```
182         dummyDB.responseArray[CONSTRUCTION_VAL].response             = 2;   // question panel 4 (2 v
    als ea.)
183         dummyDB.responseArray[SNOWPLOWING_VAL].response              = 2;   // question panel 4
184         // (1-5,6) literally, number of people
185         dummyDB.responseArray[NUMBERPEOPLE_VAL].response             = 5;   // question panel 2 (5 v
    als)
186         // (1-3); literal
187         dummyDB.responseArray[PEOPLEINFRONTSEAT_VAL].response        = 3;   // question panel 8 (3 v
    als)
188         // (1-5); scrollbar.  1 small, 5 big
189         dummyDB.responseArray[TRUCKSIZE_VAL].response                = 5;   // question panel 9 (5 v
    als)
190         dummyDB.responseArray[LIKESCHEVY_VAL].response               = 2;   // 17-24: question panel
     6 (2 vals ea.)
191         dummyDB.responseArray[LIKESDODGE_VAL].response               = 2;
192         dummyDB.responseArray[LIKESFORD_VAL].response                = 2;
193         dummyDB.responseArray[LIKESGMC_VAL].response                 = 2;
194         dummyDB.responseArray[LIKESISUZU_VAL].response               = 2;
195         dummyDB.responseArray[LIKESMAZDA_VAL].response              = 2;
196         dummyDB.responseArray[LIKESNISSAN_VAL].response             = 2;
197         dummyDB.responseArray[LIKESTOYOTA_VAL].response             = 2;
198         // 1=short, 2=long, 3=either
199         dummyDB.responseArray[BEDSIZE_VAL].response                  = 3;   // question panel 11 (3 v
    als)
200         // (2 <6; 1 6-6.5; 3 >6)
201         dummyDB.responseArray[TALLESTUSER_VAL].response              = 3;   // question panel 10 (3 v
    als)
202         dummyDB.responseArray[FOURCYLGASENGINE_VAL].response         = 2;   // question panel 12 (2 v
    als ea.)
203         dummyDB.responseArray[SIXCYLGASENGINE_VAL].response          = 2;   // question panel 12
204         dummyDB.responseArray[EIGHTCYLGASENGINE_VAL].response        = 2;   // question panel 12
205         dummyDB.responseArray[TENCYLGASENGINE_VAL].response          = 2;   // question panel 12
206         dummyDB.responseArray[EIGHTCYLDIESEL_VAL].response           = 2;   // question panel 12
207         // (1-5); 1 no comfort, 5 comfort
208         dummyDB.responseArray[COMFORT_VAL].response                  = 5;   // question panel 13 (5 v
    als)
209         // (1-5); 1 traditional, 5 futuristic
210         dummyDB.responseArray[BODYSTYLING_VAL].response              = 5;   // question panel 14 (5 v
    als)
211         dummyDB.responseArray[GREATSTERIO_VAL].response              = 2;   // 35-47: options panel
212         dummyDB.responseArray[THRDFRTHDOOR_VAL].response             = 2;
213         dummyDB.responseArray[LEATHERSEATS_VAL].response             = 2;
214         dummyDB.responseArray[CRUISECONTROL_VAL].response            = 2;
215         dummyDB.responseArray[POWERLOCK_VAL].response                = 2;
216         dummyDB.responseArray[POWERWINDOW_VAL].response              = 2;
217         dummyDB.responseArray[TILTWHEEL_VAL].response                = 2;
218         dummyDB.responseArray[TINTWINDOW_VAL].response               = 2;
219         dummyDB.responseArray[THEFTDETER_VAL].response               = 2;
220         dummyDB.responseArray[SAFETYUNIMP_VAL].response              = 2;
221         dummyDB.responseArray[BETTERTIRES_VAL].response              = 2;
222         dummyDB.responseArray[ANTILOCKBRAKES_VAL].response           = 2;
223         dummyDB.responseArray[FOURWHEELDR_VAL].response              = 2;
224         dummyDB.responseArray[AUTOMATIC_VAL].response                = 2;
225
226         return dummyDB;
227     }
228
229         private void pln(String message) {
230             //added by xing 2/27/98
231             if(debug)
232                 System.out.println(message);
233         }
234
235         private void error(String message){
236                     System.out.println("responseDB error: " + message);
237                 System.exit(0);
238         }
239 }
240
```

```
1  package Trucktown.Expert;
2
3  import Trucktown.*;
4  import java.io.*; //added by xing 2/27/98
5  //
6  //
7  // Response
8  //
9  //
10 public class Response implements Serializable{
11
12     private static final boolean DEBUG = false;
13
14         //the int response starts out life as -1 to make it easier to know if they have ever
15         //  been answered (would have used null but those are not primatives)
16         //they are in a response array (zero reserved to be nothing)
17         public int response = -1;
18         public int questionIndex;//ResponsesDB has these as static vars!!
19
20     //The sales points that are given that coorospond with each response
21     //can also be used to validate the responses matrix
22     public String[] salesFeatures;
23     public String[] salesProblems;
24     public double[] salesFeaturesWeight;
25     public double[] salesProblemsWeight;
26
27     //rank is usually based on the delta from the local baseline
28     public static final double NULL = -100000;
29     public double delta;//couldn't use -1 cuz it's a valid number
30     public double salesDelta;
31     //rank of response (usally a temp var used when looking at a particular truck)
32     public int rank = -1;
33
34         //constructor
35         public Response(int questionIndex){
36
37             this.questionIndex = questionIndex;
38             this.delta = this.NULL;
39             salesProblems = salesFeatures;
40         }
41
42         public Response(int questionIndex, String[] salesPoints){
43
44             this.questionIndex = questionIndex;
45             this.salesFeatures = salesPoints;
46             this.delta = this.NULL;
47             salesProblems = salesFeatures;
48         }
49
50         public Response(int questionIndex, String[] salesPts, String[] salesProbs) {
51             this.questionIndex = questionIndex;
52             this.salesFeatures = salesPts;
53             this.delta = this.NULL;
54             this.salesDelta = this.NULL;
55             salesProblems = salesProbs;
56         }
57
58         public Response(int questionIndex, String[] salesPts, String[] salesProbs, double[] refW, dou
   ble[] probW) {
59             this.questionIndex = questionIndex;
60             this.salesFeatures = salesPts;
61             this.delta = this.NULL;
62             this.salesDelta = this.NULL;
63             salesProblems = salesProbs;
64             salesFeaturesWeight = refW;
65             salesProblemsWeight = probW;
66         }
67
68     //can be used to get values out of the response array
69     public int getResponse()
70     {
71         return response;
72     }
73
74         public void print(){
75         //if needed, read the name of the var that reps the questionIndex
76         if (DEBUG){
77             System.out.println("response_: "+response);
78             }
79         }
80
81         private void error(String message){
82                     System.out.println("responseDB error: " + message);
83                 System.exit(0);
84         }
85
86             //added by Xing 2/25/98
87     public void writeObject(ObjectOutputStream out) throws IOException {
88         out.defaultWriteObject();
89     }
90
91     public void readObject(ObjectInputStream in) throws IOException, ClassNotFoundException {
```

```
92          in.defaultReadObject();
93      }
94
95
96  }
97
```

```
1  package Trucktown;
2
3  /*
4          Applet that will parse through our database and denote the
5          elements as objects
6  */
7  import Trucktown.Expert.*;
8  import java.awt.*;
9  import java.applet.*;
10 import java.awt.image.*;
11 import java.io.*;
12 import java.net.*;
13 import java.util.*;
14 import java.sql.*;
15 import java.rmi.*;
16 import java.rmi.server.*;
17 import Trucktown.Server.TruckDBRM;
18
19 public class TruckDB extends Object
20 {
21     boolean debug = false;
22     boolean less_trucks = false;
23     boolean remote = true;
24
25         protected Truck[] trucks;
26         protected Hashtable             truckTable;
27     protected TruckDBRM remoteDB;
28         public TruckDB(Applet theApp)
29         {
30             String url = "";
31             trucks = null;
32                 truckTable = new Hashtable();
33                 if(theApp != null) {
34                     try {
35
36                     if (remote) {
37                         url = "rmi://vandelay.mit.edu/";
38                     } else {
39                             url = "rmi://"+(theApp.getDocumentBase()).getHost()+"/";
40                 }
41
42                         remoteDB = (TruckDBRM) Naming.lookup(url+"THE_TRUCKDB");
43                         System.out.println("the remote DB obtained");
44
45                         truckTable = remoteDB.getTruckTable();
46                         System.out.println("The trucktable is retrieved");
47
48                         trucks = remoteDB.getAllTrucks();
49                         System.out.println("the trucks is retrieved");
50
51                     }
52                     catch(RemoteException re) {
53                         System.out.println("Remote Error in TruckDB:" + re);
54                     }
55                     catch(MalformedURLException mue) {
56                         //shouldn't happen.
57                         System.out.println("bad url in TruckDB");
58                     }
59                     catch(NotBoundException nbe) {
60                         //shouldn't happen.
61                         System.out.println("the name of the truckDB doesn't exits"+nbe);
62                     }
63
64                     //For Debugging Only, to be removed:
65                     //addToResponseArray(trucks[0], sampleMatrix());
66                     //this line tested to see if the addToResponse array
67                     //works or not, and uses sampleMatrix to
68                     //be the result matrix.
69                 }
70                 else {
71                         System.out.println("Error in TruckDB: parent Applet is null");
72                 }
73         }
74
75         public Truck getTruckFromName(String name)
76         {
77                 System.out.println("TruckDB:  getting truck from name = "+name);
78
79                 Truck t = (Truck)truckTable.get(name);
80
81                 if(t == null)
82                 {
83             name = name.replace('_', ' ');
84                         System.out.println("TruckDB:  getting truck from name = "+name);
85                     t = (Truck)truckTable.get(name);
86                 }
87
88                 System.out.println("Truck exists? "+(t != null));
89                 return t;
90         }
91
92     public Truck[] getAllTrucks() {
```

```
 93
 94          return trucks;
 95
 96      }
 97          public Truck GetTruck(int index)
 98          {
 99  //              if(trucks != null)
100  //              {
101                          if((index >= 0) && (index < trucks.length))
102                          {
103                                  return trucks[index];
104                          } else {
105  //                              System.out.println("TruckDB:  GetTruck called with invalid index "+i
    ndex);
106                                  return null;
107                          }
108  //              }
109
110  //              return null;
111          }
112
113
114
115          private void addToResponseArray(Truck t, double[][] result){
116              //requires: the result to be of the same length and size
117              // as the Truck.numberOfResponseArrays.
118              //modifies: this
119              //effects: updates the databases, such that each responses
120              // are increased by the amount indicated in the result.
121              try {
122              remoteDB.addToResponseArray(t, result);
123                  trucks = remoteDB.getAllTrucks();
124                  truckTable = remoteDB.getTruckTable();
125              }
126              catch(RemoteException re) {
127                  System.out.println("TruckDB:"+ re);
128              }
129          }
130
131
132      //for debug purposes only to be removed when no
133      //longer needed.
134          private double[][] sampleMatrix() {
135          //effects: creates an sample matrix that
136          // makes the certain updates.
137              double[][] matrix = new double[43][14];
138              for(int i = 0; i < 43; i++) {
139                  for(int j = 0; j < 14; j++) {
140                      matrix[i][j] = 0;
141                  }
142              }
143              matrix[1][1] = 1; //sample places where the data wants to be changed
144              matrix[3][2] = 1;
145              return matrix;
146          }
147
148
149      public void p(String s) {
150
151          if (debug)
152              System.out.print(s);
153      }
154      public void pln(String s) {
155
156          if (debug)
157              System.out.println("TruckDB: "+s);
158      }
159
160  }
161
162
163
```

```
1  package Trucktown;
2
3  /*
4  */
5
6
7  import java.awt.*;
8  import java.applet.*;
9  import java.awt.*;
10 import java.awt.image.*;
11 import java.io.*;
12 import java.net.*;
13 import java.util.*;
14
15 //added by KJGL on 2-9-98
16 import Trucktown.Expert.ResponsesDB;
17
18 public class Truck extends Object implements Serializable
19 {
20         public static final int VEHICLE_STR                                 =       0;
21     public static final int IMAGE_STR                       =   1;
22         //added by Xing for competitors.
23     public static final int COMPETITOR1 = 2;
24     public static final int COMPETITOR2 = 3;
25     public static final int COMPETITOR3 = 4;
26         public static final int TOTALSTRINGS                                =       5;
27
28         //this is not part of the string array
29         public String segmentURL = "";
30
31         //used by elimination filter in expert decision
32         public int eliminationCount = 0;
33
34         public static final int MSRP_VAL                                    =       0;
35         public static final int INVOICEPRICE_VAL                            =       1;
36         public static final int DEALERHOLDBACK_VAL                          =       2;
37         public static final int DESTINATION_VAL                             =       3;
38         public static final int ORIGIN_VAL                                  =       4;
39         public static final int MANUFACTURER_VAL                            =       5;
40         public static final int TWOWD_4WD_VAL                               =       6;
41         public static final int EXTENDEDCAB_VAL                             =       7;
42         public static final int FUELECONOMY_CITY_VAL                =       8;
43         public static final int FUELECONOMY_HIGHWAY_VAL             =       9;
44         public static final int RELIABILITY_VAL                             =       10;
45         public static final int DEPENDABILITY_VAL                           =       11;
46         public static final int SAFETY_VAL                                  =       12;
47         public static final int SIZE_VAL                                    =       13;
48         public static final int BODYSTYLE_VAL                               =       14;
49         public static final int BEDLENGTH_VAL                               =       15;
50         public static final int BEDUPGRADEPRICE_VAL                         =       16;
51         public static final int BASETRANSMISSION_VAL                =       17;
52         public static final int SPEEDS_VAL                                  =       18;
53         public static final int FUELCAPACITY_VAL                            =       19;
54         public static final int STANDARDHP_VAL                              =       20;
55         public static final int ENGINESIZE_VAL                              =       21;
56         public static final int HPV4_VAL                                    =       22;
57         public static final int UPGRADEPRICELISTV4_VAL              =       23;
58         public static final int UPGRADEPRICEINVOICEV4_VAL           =       24;
59         public static final int ENGINESIZEV6_VAL                            =       25;
60         public static final int HPV6_VAL                                    =       26;
61         public static final int UPGRADEPRICELISTV6_VAL              =       27;
62         public static final int UPGRADEPRICEINVOICEV6_VAL           =       28;
63         public static final int ENGINESIZEV6_OP2_VAL                =       29;
64         public static final int HPV6_OP2_VAL                                =       30;
65         public static final int UPGRADEPRICELISTV6_OP2_VAL          =       31;
66         public static final int UPGRADEPRICEINVOICEV6_OP2_VAL   =       32;
67         public static final int ENGINESIZEV8_VAL                            =       33;
68         public static final int HPV8_VAL                                    =       34;
69         public static final int UPGRADEPRICELISTV8_VAL              =       35;
70         public static final int UPGRADEPRICEINVOICEV8_VAL           =       36;
71         public static final int ENGINESIZEV8_OP2_VAL                =       37;
72         public static final int HPV8_OP2_VAL                                =       38;
73         public static final int UPGRADEPRICELISTV8_OP2_VAL          =       39;
74         public static final int UPGRADEPRICEINVOICEV8_OP2_VAL   =       40;
75         public static final int V10ENGINEOPTION_VAL                         =       41;
76         public static final int HPV10_VAL                                   =       42;
77         public static final int UPGRADEPRICELISTV10_VAL             =       43;
78         public static final int UPGRADEPRICEINVOICEV10_VAL          =       44;
79         public static final int DIESELENGINEOPTION_VAL              =       45;
80         public static final int NUMCYL_VAL                                  =       46;
81         public static final int HPDIESEL_VAL                                =       47;
82         public static final int UPGRADEPRICELISTDIESEL_VAL          =       48;
83         public static final int UPGRADEPRICEINVOICEDIESEL_VAL   =       49;
84         public static final int PAYLOAD_VAL                                 =       50;
85         public static final int TOWINGCAPACITY_VAL                          =       51;
86         public static final int ACLIST_VAL                                  =       52;
87         public static final int ACINVOICE_VAL                               =       53;
88         public static final int WARRANTY_VAL                                =       54;
89         public static final int WHEELBASE_VAL                               =       55;
90
91         // Andy: new truck val's start here
92         public static final int AUTOLIST_VAL                                =       56;
```

```
93          public static final int AUTOINVOICE_VAL                                 =       57;
94          public static final int AIRBAGLIST_VAL                                  =       58;
95          public static final int AIRBAGINVOICE_VAL                               =       59;
96          public static final int SOUNDLIST_VAL                           =       60;
97          public static final int SOUNDINVOICE_VAL                        =       61;
98          public static final int CRUISELIST_VAL                              =   62;
99          public static final int CRUISEINVOICE_VAL                           =   63;
100         public static final int LEATHERLIST_VAL                                 =       64;
101         public static final int LEATHERINVOICE_VAL                              =       65;
102         public static final int POWERLOCKLIST_VAL                               =       66;
103         public static final int POWERLOCKINVOICE_VAL                    =       67;
104         public static final int POWERWINDOWLIST_VAL                         =   68;
105         public static final int POWERWINDOWINVOICE_VAL                  =       69;
106         public static final int THEFTDETERLIST_VAL                              =       70;
107         public static final int THEFTDETERINVOICE_VAL                   =       71;
108         public static final int TILTLIST_VAL                                    =       72;
109         public static final int TILTINVOICE_VAL                                 =       73;
110         public static final int TINTEDLIST_VAL                                  =       74;
111         public static final int TINTEDINVOICE_VAL                               =       75;
112         public static final int TIRESLIST_VAL                           =       76;
113         public static final int TIRESINVOICE_VAL                        =       77;
114     // new val's end
115
116         public static final int LENGTH_VAL                                                      =       78;
117         public static final int HEIGHT_VAL                                                      =       79;
118         public static final int WIDTH_VAL                                                       =       80;
119
120         // these are no longer needed.
121         public static final int MSRPCORRECT_VAL                                 =       81;
122         public static final int FUELECONOMY_VAL                         =       82;
123         public static final int DEPENDABILITYCORRECT_VAL                =       83;
124         public static final int DURABILITY_VAL                              =   84;
125         public static final int HORSEPOWER_VAL                          =       85;
126
127         public static final int SAFETYCORRECT_VAL                               =       86;
128
129     // there are 155 Bayesian vals, for a total of 250 doubles
130         public static final int TOTALVALUES                                                     =       250;
131
132         public String[]                 strings;//changed from protected to public by Xing 2/16/98
133         public double[]                 numbers;//needed because the server need to access these
134                                         //fields through another package.
135
136         /*needed to make bayesian decisions!!
137         numberArray is needed to go to a truely learning network.
138         may be a really good idea to use constants to remind what the hell is going on
139         normalizedResponses is the numberOfResponseArrays massaged into a form more
140         //easily used by the bayesian expert,
141         !!!these arrays do not use the 0 possition, to better match up with questionaire
142                 aka. 1-5 doesn't have to be translated into 0-4!!!
143         */
144         //to be calculated over and over again!
145         public double prioriProb;
146         //isNormalized is true when the matrix is filled in, false when it's empty
147         public boolean isNormalized = false;
148         //matrix of probs that the question in this truck whatever way
149         // ART: don't need to fill this out.
150         public double[][] normalizedResponseArrays;
151
152         //from chip allocations to first a priori's (intermediate values)
153         public double utility;
154         public double choiceValue;
155
156         /*
157             These values are to be filled w/ numbers in database.
158         */
159
160         //this is posteriori info. int[question_n][response_n] to be filled be for expert
161         //ART: to be filled w/ numbers in database
162         public double[][] numberOfResponseArrays;
163         //normalized values to calculate the utility from
164         public double MRSP = 20;
165         public double FuelEconomy = 20;
166         public double Dependability = 20;
167         public double Safety = 20;
168         public double HorsePower = 20;
169
170         // this contains the segment data
171         // it's got 8 vals, booleans
172         public boolean[] inSegment;
173     //added by KJGL on 2.21.98
174     //segment reason for this truck being recommended (added when the truck is run
175         private String segmentSalesFeature = null;
176
177     //added by KJGL on 2.21.98
178     //bayesian reasons for this truck being recommended (added when the truck is run
179     //through the Bayesian Reasoner (the most important reason being listed first)
180     private String[] bayesianSalesFeatures;
181     //added by KJGL on 3.25.98
182     private String[] bayesianSalesProblems;
183
184     //added by KJGL on 3.16.98
```

```
185     public static final int BAYESIAN_REASON = 1;
186     public static final int SEGMENT_REASON = 2;
187     public static final int LAST_TRUCK_REASON = 3;
188     public static final String lastTruckString
189             = "This truck is similar to the last truck that you perchased. It is here to be able to
    compair to the other trucks.";
190
191     public int recommendationReason = 0;
192
193     public Magazine the_magazine;
194     public Comment the_comment;
195         /*end v2 stuff*/
196
197         public Truck()
198         {
199                 if(!InitArrays())
200                 {
201                     throw new InternalError("Couldn't intialize a truck object.");
202                 //System.out.println("Couldn't initialize a truck object.");
203                 }
204         }
205
206
207         public boolean
208         InitArrays()
209         {
210                 strings = new String[TOTALSTRINGS];
211                 numbers = new double[TOTALVALUES];
212                 inSegment = new boolean[8];
213
214                 return ((strings != null) && (numbers != null));
215         }
216
217
218     //
219         // tells whether this truck is in a specified segment (0-aligned)
220         public boolean
221         isInSegment(int index) {
222                 // check if index is within bounds
223                 if((index >= 0) && (index < 8) && (inSegment != null)) {
224                         // return the value
225                         return inSegment[index];
226                 }
227                 return false;
228         }
229
230         // sets whether this truck is in the specified segment (0-aligned)
231         public void
232         setInSegment(int index,boolean value) {
233                 // check if index is within bounds
234                 if((index >= 0) && (index < 8) && (inSegment != null)) {
235                         // set the value
236                         inSegment[index] = value;
237                 }
238         }
239
240         //added by KJGL on 2.21.98
241         public void setSegmentSalesFeature(String segmentSalesFeature){
242             this.segmentSalesFeature = segmentSalesFeature;
243         }
244         //added by KJGL on 2.21.98
245         public String getSegmentSalesFeature(){
246             return this.segmentSalesFeature;
247         }
248         //added by KJGL on 2.21.98
249         public void setBayesianSalesFeatures(String[] bayesianSalesFeatures){
250             this.bayesianSalesFeatures = bayesianSalesFeatures;
251         }
252         //added by KJGL on 2.21.98
253         public String[] getBayesianSalesFeatures(){
254             return this.bayesianSalesFeatures;
255         }
256         //added by KJGL on 3.25.98
257         public void setBayesianSalesProblems(String[] bayesianSalesProblems){
258             this.bayesianSalesProblems = bayesianSalesProblems;
259         }
260         //added by KJGL on 3.25.98
261         public String[] getBayesianSalesProblems(){
262             return this.bayesianSalesProblems;
263         }
264
265     //added by KJGL on 2-9-98
266     //validates response matrix and then sees if the responseDB matches up with it
267         public void validateResponseData(ResponsesDB responsesDB){
268             System.out.println("Took a look at " + this.GetString(Truck.VEHICLE_STR));
269             //validate that the numberOfResponseArrays has correct looking data
270             //the array of arrays is the same length as responseArray
271         try{
272             if (numberOfResponseArrays.length != responsesDB.responseArray.length){
273                 error("numberOfResponseArrays and responseArray are not the same length");
274             }
275         }
```

```
276                 catch(NullPointerException e){
277                     error("problem with the reading the numberOfResponseArrays");
278                 }
279
280             //make sure the data in the arrays is non-negative (zero or above)
281             int questionIndex = 0;
282             int responseIndex = 0;
283         try{
284             for ( questionIndex = 1; questionIndex < numberOfResponseArrays.length; questionIndex++)
    {
285                 //going through all of the response arrays
286                 for ( responseIndex = 1; responseIndex < numberOfResponseArrays[questionIndex].lengt
    h; responseIndex++){
287                     //to see if the data is negative
288                     if (numberOfResponseArrays[questionIndex][responseIndex] < 0){
289                         //it's negative so it's invalid
290                         error("negative value in number of response arrays");
291                     }
292                 }
293             }
294         }
295                 catch(NullPointerException e){
296                     error("problem with the reading the numberOfResponseArrays ["+questionIndex+"] [
    "+responseIndex+"]");
297                 }
298
299             //make sure that the responseDB data coorisponds to a point in the second array
300             for (int i = 1; i<responsesDB.responseArray.length; i++){
301             try{
302                 if (responsesDB.responseArray[i].response != -1){
303                     //a response was made, so see if it references an actual place in the response m
    atrix
304                     if (numberOfResponseArrays[i][responsesDB.responseArray[i].response] < 0){
305                         //referenced data that was negative
306                         error("negative value in number of response arrays");
307                     }
308                 }
309             }
310                 catch(NullPointerException e){
311                     error("problem with the reading the numberOfResponseArrays ["+i+"] []");
312                 }
313                 }
314             //see that all the arrays begin with null (if there are still problems)
315     }
316
317         private void error(String message){
318                 System.out.println("this.GetString(Truck.VEHICLE_STR) error: " + message);
319             System.exit(0);
320         }
321
322         //*******Irene's lovely additions***************
323
324         public Magazine[] getMags()
325         {
326             // this function looks at the database information to find the location of some
327             // magazine articles about the truck.  It will then parse each article into a
328             // Magazine object and return those objects as an array.
329
330             // as the magazine locations (or actual text) have not been entered into the server,
331             // I will now make some dummy test Magazines.
332             Magazine Mags[] = new Magazine[1];
333             Mags[0] = the_magazine;
334
335             /*  Magazine Mag = new Magazine("Joe Shmoe", "The Chevy 1800 VWX is a really super car.
    " +
336                 "I think I'll go out and buy one today because it is such a nice rosey color.  It" +
337                 " makes me think of cherries and fruit punch.", "This Cool Magazine", "10/11/78",
338                 "The Chevy and Why Joe Likes It");
339
340             Mags[0] = Mag;
341
342             Mag = new Magazine("Lucy Shmeeler", "So, here I talk about some car... still taling " +
343                 "about a car... But to be completely frank, I don't find cars very interesting.  " +
344                 "I think they are too annoying to take care of.  You always have to be paying to" +
345                 " have them taken to the shop.", "Yo Yo Magazine", "12/12/98", "Lucy and the Car");
346
347             Mags[1] = Mag;
348         */
349             return Mags;
350         }
351
352         public Comment[] getComs()
353         {
354             // this function will get comment information from the database and use them to create
355             // Comment objects and return those objects as an array.
356
357             // as the comment have not been entered into the server, I will now make some dummy
358             //test Magazines.
359
360             Comment Coms[] = new Comment[1];
361             Coms[0]=the_comment;
362
```

```
363             /*Comment Com = new Comment("Charlie Brown", "I like cars.  I'd better say something " +
364             "else, so that I can be sure this format will wrap the text correctly.");
365
366             Coms[0] = Com;
367
368             Com = new Comment("Dick Sweat", "Hey, oh, hey, now I sound like MC Hammer.  La la " +
369             "la... I hope no sponsor ever reads this... yo yo.");
370
371             Coms[1] = Com;
372         */
373             return Coms;
374         }
375
376         //**********end of Irene's lovely additions :(  *********************
377
378         public String
379         GetString(int index)
380         {
381                 if((index >= 0) && (index < TOTALSTRINGS) && (strings != null))
382                 {
383                         return strings[index];
384                 }
385
386                 return null;
387         }
388
389         public void
390         PutString(int index, String value)
391         {
392                 if((index >= 0) && (index < TOTALSTRINGS) && (strings != null))
393                 {
394                         strings[index] = new String(value);
395                         if (index == 0 ) {
396                                 strings[index] = strings[index].replace('_', ' ');
397         //                      System.out.println("replacing _'s: "+strings[index]);
398                         }
399                 }
400         }
401
402         public double GetValue(int index)
403         {
404                 if((index >= 0) && (index < TOTALVALUES) && (numbers != null))
405                 {
406                         return numbers[index];
407                 }
408
409                 return 0.0;
410         }
411
412         public void
413         PutValue(int index, double value)
414         {
415                 if((index >= 0) && (index < TOTALVALUES) && (numbers != null))
416                 {
417                         numbers[index] = value;
418                 }
419         }
420
421         public String getResponseMatrixString(){
422             String result= "";
423         for(int i=1; i<numberOfResponseArrays.length; i++) {
424             for(int j=1; j<numberOfResponseArrays[i].length; j++) {
425                 result = result+numberOfResponseArrays[i][j] + " ";
426             }
427             result = result + "\n";
428         }
429
430                 return result;
431         }
432
433
434         public String toString() {
435
436                 return "";
437         }
438
439         private void writeObject(ObjectOutputStream out) throws IOException {
440             out.defaultWriteObject();
441         }
442
443         private void readObject(ObjectInputStream in) throws IOException, ClassNotFoundException{
444             in.defaultReadObject();
445         }
446 }
447
448
```

```
 1 package Trucktown.Expert;
 2 import Trucktown.*;
 3 import java.awt.*;
 4
 5
 6 public class ExpertHistory
 7 {
 8     //starts life this way
 9     public boolean questionConfigPanelShown = false;
10     //is true if the logic expert said that the expert is thinking something
11     public boolean expertIsThinkingSomething = false;
12
13     // these are generically used to access checkboxes in the hacked arrays below
14     static final int CHECK1 =   0;
15     static final int CHECK2 =   1;
16     static final int CHECK3 =   2;
17     static final int CHECK4 =   3;
18     static final int CHECK5 =   4;
19     static final int CHECK6 =   5;
20
21     // **temporary** these are specific to the option config panel.  won't be needed once fixed
22     static final int OPTION_CHECK10 =  0;
23     static final int OPTION_CHECK12 =  1;
24     static final int OPTION_CHECK13 =  2;
25
26     // these are for question 15 either boxes
27     static final int QUES15_CHECK3 = 0;
28     static final int QUES15_CHECK6 = 1;
29
30     // permanent array for question 12 becuase of the wacked out I don't know button
31     public double ques12Boxes[] = {-1, -1, -1, -1, -1, -1};
32
33     // temporary hack to save missing option config panel
34     public double optionConfigBoxes[] = {-1, -1, -1};
35
36     public double ques15Boxes[] = {-1, -1};
37
38         //constructor
39         public ExpertHistory()
40         {
41         }
42 }
```

E. Truck Utility Database

Vehicle	MSRP	Fuel_Economy	Dependability	Horsepower	Safty
Chevy_C1500_(2WD)	-0.59	0.37	-0.73	0.54	0.79
Chevy_C1500_Extended_(2WD)	-0.02	0.37	-0.73	0.54	-0.34
Chevy_C2500_(2WD)	-0.19	-0.19	-0.73	0.54	0.35
Chevy_C2500_Extended_(2WD)	0.52	-0.19	-0.73	1.17	0.25
Chevy_C3500_(2WD)	0.08	-0.76	-0.73	1.17	1.34
Chevy_C3500_Crew_(2WD)	0.93	-1.04	-0.73	1.17	0.81
Chevy_C3500_Extended_(2WD)	1.01	-0.47	-0.73	1.17	1.69
Chevy_K1500_(4WD)	0.21	-0.19	-0.73	0.54	-0.51
Chevy_K1500_Extended_(4WD)	0.74	-0.19	-0.73	0.54	-0.32
Chevy_K2500_(4WD)	0.79	-0.47	-0.73	1.17	1.15
Chevy_K2500_Extended_(4WD)	1.32	-0.47	-0.73	1.17	0.61
Chevy_K3500_(4WD)	0.9	-0.76	-0.73	1.17	1.19
Chevy_K3500_Crew_(4WD)	1.79	-1.04	-0.73	1.17	0.69
Chevy_K3500_Extended_(4WD)	1.77	-1.04	-0.73	1.17	1.35
Chevy_S-10_(2WD)	-1.74	2.07	1.42	-0.64	-1.93
Chevy_S-10_(4WD)	-0.47	0.37	1.42	-0.64	-1.93
Chevy_S-10_Extended_(2WD)	-0.96	2.07	1.42	-0.64	-1.93
Chevy_S-10_Extended_(4WD)	0.29	0.37	1.42	-0.64	-1.93
Dodge_Dakota_(2WD)	-1.46	1.5	1.65	0.08	0.09
Dodge_Dakota_(4WD)	-0.05	-0.19	1.65	0.08	-0.37
Dodge_Dakota_Club_Cab_(2WD)	-0.46	0.09	1.65	0.08	-0.37
Dodge_Dakota_Club_Cab_(4WD)	0.43	-0.19	1.65	0.08	-0.53
Dodge_Ram_1500_(2WD)	-1.09	-0.19	1.08	0.17	-0.37
Dodge_Ram_1500_(4WD)	0.25	-0.76	1.08	0.17	-0.47
Dodge_Ram_1500_Club_(2WD)	0.12	-0.47	1.08	0.17	-0.51
Dodge_Ram_1500_Club_(4WD)	0.96	-0.76	1.08	0.17	-0.81
Dodge_Ram_2500_(2WD)	0.18	-1.04	1.08	1.35	1.21
Dodge_Ram_2500_(4WD)	1.02	-1.04	1.08	1.35	0.85
Dodge_Ram_2500_Club_(2WD)	0.73	-1.04	1.08	1.35	1.04
Dodge_Ram_2500_Club_(4WD)	1.54	-1.04	1.08	1.35	0.84
Dodge_Ram_3500_(2WD)	0.58	-1.04	1.08	1.35	2.15
Dodge_Ram_3500_(4WD)	1.27	-1.04	1.08	1.35	1.87
Dodge_Ram_3500_Club_(2WD)	1.22	-1.04	1.08	1.35	1.89
Dodge_Ram_3500_Club_(4WD)	1.91	-1.04	1.08	1.35	2
Ford_F150_(2WD)	-1	0.37	0.4	0.17	-0.61
Ford_F150_(4WD)	-0.04	0.09	0.4	0.17	-0.56
Ford_F150_Extended_(2WD)	-0.41	0.37	0.4	0.17	-0.41
Ford_F150_Extended_(4WD)	0.63	-0.19	0.4	0.17	-0.75
Ford_F250_(2WD)	-0.61	0.09	0.4	0.35	1.24
Ford_F250_(4WD)	0.71	-0.19	0.4	0.35	0.86
Ford_F250_Extended_(2WD)	0.44	0.09	0.4	0.35	1.05
Ford_F250_Extended_(4WD)	1.29	-0.19	0.4	0.35	0.82
Ford_F350_(2WD)	-0.03	-0.76	0.4	0.35	1.42
Ford_F350_(4WD)	0.8	-1.32	0.4	0.35	1.2
Ford_F350_Crew_Cab_(2WD)	0.87	-0.76	0.4	0.35	1.17
Ford_F350_Crew_Cab_(4WD)	1.7	-1.32	0.4	0.35	0.82
Ford_F350_Extended_(2WD)	0.81	-1.04	0.4	0.35	1.69
Ford_Ranger_(2WD)	-1.94	1.78	-1.64	-1.19	-0.65
Ford_Ranger_(4WD)	-0.47	0.65	-1.64	-1.19	-0.72

E. Truck Utility Database

Ford_Ranger_Extended_(2WD)	-1.03	0.94	-1.64	-1.19	-0.72
Ford_Ranger_Extended_(4WD)	-0.03	0.65	-1.64	-1.19	-0.8
GMC_Sierra_C1500_(2WD)	-0.95	0.09	0.35	0.54	-0.19
GMC_Sierra_C1500_Extended_(2WD)	-0	0.37	0.35	0.54	-0.34
GMC_Sierra_C2500_(2WD)	-0.17	-0.19	0.35	0.54	0.33
GMC_Sierra_C2500_Extended_(2WD)	0.5	-0.19	0.35	0.54	0.86
GMC_Sierra_C3500_(2WD)	0.1	-0.76	0.35	1.17	1.28
GMC_Sierra_C3500_Crew_Cab_(2WD)	0.95	-0.76	0.35	1.17	0.74
GMC_Sierra_C3500_Extended_(2WD)	1.03	-0.76	0.35	1.17	0.56
GMC_Sierra_K1500_(4WD)	0.05	-0.19	0.35	0.54	-0.51
GMC_Sierra_K1500_Extended_(4WD)	0.76	-0.19	0.35	0.08	-0.32
GMC_Sierra_K2500_(4WD)	0.81	-0.47	0.35	1.17	0.75
GMC_Sierra_K2500_Extended_(4WD)	1.34	-0.47	0.35	1.17	0.53
GMC_Sierra_K3500_(4WD)	0.92	-0.76	0.35	1.17	1.42
GMC_Sierra_K3500_Crew_Cab_(4WD)	1.81	-0.76	0.35	1.17	0.46
GMC_Sierra_K3500_Extended_(4WD)	1.79	-0.76	0.35	1.17	0.26
GMC_Sonoma_(2WD)	-1.76	2.07	1.03	-0.83	-0.99
GMC_Sonoma_(4WD)	-0.47	0.37	1.03	-0.64	-0.66
GMC_Sonoma_Extended_(2WD)	-0.92	2.07	1.03	-0.83	-1
GMC_Sonoma_Extended_(4WD)	0.36	0.37	1.03	-0.64	-1.22
Isuzu_Hombre	-1.85	2.07	1.03	-1.95	-1.04
Isuzu_Hombre_Extended	-1.03	2.07	1.03	-0.92	-1.03
Mazda_B2300_(2WD)	-2.07	1.78	0.4	-2.06	-0.96
Mazda_B2300_Extended_(2WD)	-0.93	1.78	0.4	-2.06	-0.72
Mazda_B4000_(4WD)	-0.47	0.37	0.4	-1.19	-0.96
Mazda_B4000_Extended_(4WD)	0.04	0.37	0.4	-1.19	-0.8
Nissan_Truck_(2WD)	-1.93	1.78	-1.69	-1.66	-0.84
Nissan_Truck_(4WD)	-0.52	0.65	-1.69	-1.66	-0.84
Nissan_Truck_Extended_(2WD)	-0.94	1.78	-1.69	-1.66	-0.84
Nissan_Truck_Extended_(4WD)	-0.01	0.65	-1.69	-1.66	-0.84
Toyota_T100_(2WD)	-0.94	1.22	-0.39	-1.37	-0.62
Toyota_T100_Extended_(2WD)	0.12	0.37	-0.39	-0.64	-0.26
Toyota_T100_Extended_(4WD)	1.15	0.37	-0.39	-0.64	-0.38
Toyota_Tacoma_(2WD)	-1.6	2.07	-1.75	-1.52	-0.62
Toyota_Tacoma_(4WD)	-0.28	0.94	-1.75	-0.64	-0.44
Toyota_Tacoma_Extended_(2WD)	-1.01	2.07	-1.75	-0.64	-0.57
Toyota_Tacoma_Extended_(4WD)	0.11	0.94	-1.75	-0.64	-0.46

F. Recommendation Database

Vehicle	Chevy_C1500_(2WD)	Chevy_C1500_Extended_(2WD)	Chevy_C2500_(2WD)	Chevy_C2500_Extended_(2WD)	Chevy_C3500_(2WD)	Chevy_C3500_Crew_(2WD)	Chevy_C3500_Extended_(2WD)	Chevy_K1500_(4WD)	Chevy_K1500_Extended_(4WD)	Chevy_K2500_(4WD)	Chevy_K2500_Extended_(4WD)	Chevy_K3500_(4WD)	Chevy_K3500_Crew_(4WD)	Chevy_K3500_Extended_(4WD)	Chevy_S-10_(2WD)
compact	20	20	20	20	20	20	20	20	20	20	20	20	20	20	60
full	60	60	60	60	60	60	60	60	60	60	60	60	60	60	20
either	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20
Price_10_12K	10	1	1	1	1	1	1	1	1	1	1	1	1	1	60
Price_12_14K	20	1	9	1	1	1	1	1	1	1	1	1	1	1	20
Price_14_16K	40	8	20	1	8	1	1	8	1	1	1	1	1	1	15
Price_16_18K	20	20	40	7	20	7	7	20	7	7	1	7	1	1	1
Price_18_20K	7	40	20	20	40	20	20	40	20	20	6	20	1	1	1
Price_20_22K	1	20	8	40	20	40	40	20	40	40	20	40	15	15	1
Price_22_24K	1	9	1	20	9	20	20	9	20	20	40	20	20	20	1
Price_Over_24K	1	1	1	10	1	10	10	1	10	10	20	10	60	60	1
Off_road_no	65	65	65	65	65	65	65	35	35	35	35	35	35	35	65
Off_road_yes	35	35	35	35	35	35	35	65	65	65	65	65	65	65	35
Towing_no	50	50	50	50	50	50	50	30	30	30	30	30	30	30	70
Towing_yes	50	50	50	50	50	50	50	70	70	70	70	70	70	70	30
Hauling_no	35	35	35	35	35	35	35	35	35	35	35	35	35	35	50
Hauling_yes	65	65	65	65	65	65	65	65	65	65	65	65	65	65	50
Construction_no	35	35	35	35	35	35	35	35	35	35	35	35	35	35	50
Construction_yes	65	65	65	65	65	65	65	65	65	65	65	65	65	65	50
Plowing_Snow_no	35	35	35	35	35	35	35	35	35	35	35	35	35	35	50
Plowing_Snow_yes	65	65	65	65	65	65	65	65	65	65	65	65	65	65	50
Passengers_1	25	10	25	10	25	10	10	25	10	25	10	25	10	10	25
Passengers_2	25	15	25	15	25	15	15	25	15	25	15	25	15	15	25
Passengers_3	25	25	25	25	25	25	25	25	25	25	25	25	25	25	25
Passengers_4	15	25	15	25	15	25	25	15	25	15	25	15	25	25	15
Passengers_5	10	25	10	25	10	25	25	10	25	10	25	10	25	25	10
Passengers_front_seat_1	40	40	40	40	40	40	40	40	40	40	40	40	40	40	40
Passengers_front_seat_2	40	40	40	40	40	40	40	40	40	40	40	40	40	40	40
Passengers_front_seat_3	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20
Biggest_available_1	10	10	10	10	10	10	10	10	10	10	10	10	10	10	25
Biggest_available_2	15	15	15	15	15	15	15	15	15	15	15	15	15	15	25
Biggest_available_3	25	25	25	25	25	25	25	25	25	25	25	25	25	25	25
Biggest_available_4	25	25	25	25	25	25	25	25	25	25	25	25	25	25	15
Chevy_no	25	25	25	25	25	25	25	25	25	25	25	25	25	25	25
Chevy_yes	75	75	75	75	75	75	75	75	75	75	75	75	75	75	75
Dodge_no	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50
Dodge_yes	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50
Ford_no	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50
Ford_yes	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50
GMC_no	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50
GMC_yes	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50
Isuzu_no	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50
Isuzu_yes	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50
Mazda_no	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50
Mazda_yes	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50
Nissan_no	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50
Nissan_yes	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50
Toyota_no	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50
Toyota_yes	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50
Bed_legnth_short	35	35	35	35	35	35	35	35	35	35	35	35	35	35	50
Bed_length-long	50	50	50	50	50	50	50	50	50	50	50	50	50	50	35
height_Under_6	30	30	30	30	30	30	30	30	30	30	30	30	30	30	25
height_between_6_65	40	40	40	40	40	40	40	40	40	40	40	40	40	40	45
height_over_65	30	30	30	30	30	30	30	30	30	30	30	30	30	30	45
engine_size_4_no	80	80	80	80	80	80	80	80	80	80	80	80	80	80	20
engine_size_4_yes	20	20	20	20	20	20	20	20	20	20	20	20	20	20	80
engine_size_6_no	20	20	80	80	80	80	80	20	20	80	80	80	80	80	20
engine_size_6_yes	80	80	20	20	20	20	20	80	80	20	20	20	20	20	80
engine_size_8_no	20	20	20	20	20	20	20	20	20	20	20	20	20	20	80
engine_size_8_yes	80	80	80	80	80	80	80	80	80	80	80	80	80	80	20
engine_size_10_no	80	80	80	80	80	80	80	80	80	80	80	80	80	80	80
engine_size_10_yes	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20
engine_size_8_diesel_no	80	80	20	20	20	20	20	80	20	20	20	20	20	20	80
engine_size_8_diesel_yes	20	20	80	80	80	80	80	20	80	80	80	80	80	80	20
Big_quiet_comfortable_1	10	10	10	10	10	10	10	10	10	10	10	10	10	10	25
Big_quiet_comfortable_2	15	15	15	15	15	15	15	15	15	15	15	15	15	15	25
Big_quiet_comfortable_3	25	25	25	25	25	25	25	25	25	25	25	25	25	25	25
Big_quiet_comfortable_4	25	25	25	25	25	25	25	25	25	25	25	25	25	25	15
Big_quiet_comfortable_5	25	25	25	25	25	25	25	25	25	25	25	25	25	25	10
traditional_futuristic_1	25	25	25	25	25	25	25	25	25	25	25	25	25	25	25
traditional_futuristic_2	25	25	25	25	25	25	25	25	25	25	25	25	25	25	25
traditional_futuristic_3	25	25	25	25	25	25	25	25	25	25	25	25	25	25	25
traditional_futuristic_4	15	15	15	15	15	15	15	15	15	15	15	15	15	15	15
traditional_futuristic_5	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10
Premium_Sterio_no	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50
Premium_Sterio_yes	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50
third_fourth_door_no	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50
third_fourth_door_yes	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50
leather_seats_no	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100
leather_seats_yes	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
cruise_control_no	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50
cruise_control_yes	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50
power_locks_no	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50
power_locks_yes	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50
power_windows_no	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50
power_windows_yes	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50
tilt_wheel_no	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50
tilt_wheel_yes	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50
tinted_windows_no	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50
tinted_windows_yes	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50
theft_deterent_no	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50
theft_deterent_yes	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50
safety_features_no	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50
safety_features_yes	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50
better_tires_no	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50
better_tires_yes	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50
Antilock_no	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50
Antilock_yes	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50
2wd	70	70	70	70	70	70	70	30	30	30	30	30	30	30	70
4wd	30	30	30	30	30	30	30	70	70	70	70	70	70	70	30
Automatic_no	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50
Automatic_yes	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50

Chevy_S-10_(4WD)	Chevy_S-10_Extended_(2WD)	Chevy_S-10_Extended_(4WD)	Dodge_Dakota_(2WD)	Dodge_Dakota_(4WD)	Dodge_Dakota_Club_Cab_(2WD)	Dodge_Dakota_Club_Cab_(4WD)	Dodge_Ram_1500_(2WD)	Dodge_Ram_1500_(4WD)	Dodge_Ram_1500_Club_(2WD)	Dodge_Ram_1500_Club_(4WD)	Dodge_Ram_2500_(2WD)	Dodge_Ram_2500_(4WD)	Dodge_Ram_2500_Club_(2WD)	Dodge_Ram_2500_Club_(4WD)	Dodge_Ram_3500_(2WD)	Dodge_Ram_3500_(4WD)	Dodge_Ram_3500_Club_(2WD)
60	60	60	60	60	60	60	20	20	20	20	20	20	20	20	20	20	20
20	20	20	20	20	20	20	60	60	60	60	60	60	60	60	60	60	60
20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20
1	10	1	20	1	1	1	10	1	1	1	1	1	1	1	1	1	1
9	20	1	40	9	9	1	20	1	1	1	1	1	1	1	1	1	1
20	40	8	20	20	20	8	40	8	8	1	8	1	1	1	1	1	1
40	20	20	10	40	40	20	20	20	20	7	20	7	7	1	7	1	1
20	7	40	7	20	20	40	7	40	40	20	40	20	20	16	20	6	16
8	1	20	1	8	8	20	1	20	20	40	20	40	40	20	40	20	20
1	1	9	1	1	1	9	1	9	9	20	9	20	20	40	20	40	40
1	1	1	1	1	1	1	1	1	1	10	1	10	10	20	10	20	20
35	65	35	65	35	65	35	65	35	65	35	65	35	65	35	65	35	65
65	35	65	35	65	35	65	35	65	35	65	35	65	35	65	35	65	35
30	70	30	70	30	70	30	50	30	50	30	50	30	50	30	50	30	50
70	30	70	30	70	30	70	50	70	50	70	50	70	50	70	50	70	50
50	50	50	50	50	50	50	35	35	35	35	35	35	35	35	35	35	35
50	50	50	50	50	50	50	65	65	65	65	65	65	65	65	65	65	65
50	50	50	50	50	50	50	35	35	35	35	35	35	35	35	35	35	35
50	50	50	50	50	50	50	65	65	65	65	65	65	65	65	65	65	65
50	50	50	50	50	50	50	35	35	35	35	35	35	35	35	35	35	35
50	50	50	50	50	50	50	65	65	65	65	65	65	65	65	65	65	65
25	10	10	25	25	10	10	25	25	10	10	25	25	10	10	25	25	10
25	15	15	25	25	15	15	25	25	15	15	25	25	15	15	25	25	15
25	25	25	25	25	25	25	25	25	25	25	25	25	25	25	25	25	25
15	25	25	15	15	25	25	15	15	25	25	15	15	25	25	15	15	25
10	25	25	10	10	25	25	10	10	25	25	10	10	25	25	10	10	25
40	20	40	40	40	20	40	40	40	40	40	40	40	40	40	40	40	40
40	40	40	40	40	40	40	40	40	40	40	40	40	40	40	40	40	40
20	40	20	20	20	40	20	20	20	20	20	20	20	20	20	20	20	20
25	25	25	25	25	25	25	10	10	10	10	10	10	10	10	10	10	10
25	25	25	25	25	25	25	15	15	15	15	15	15	15	15	15	15	15
25	25	25	25	25	25	25	25	25	25	25	25	25	25	25	25	25	25
15	15	15	15	15	15	15	25	25	25	25	25	25	25	25	25	25	25
25	25	25	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50
75	75	75	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50
50	50	50	25	25	25	25	25	25	25	25	25	25	25	25	25	25	25
50	50	50	75	75	75	75	75	75	75	75	75	75	75	75	75	75	75
50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50
50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50
50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50
50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50
50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50
50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50
50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50
50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50
50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50
50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50
50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50
50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50
50	50	50	50	50	50	50	35	35	35	35	35	35	35	35	35	35	35
35	35	35	35	35	35	35	50	50	50	50	50	50	50	50	50	50	50
25	25	25	25	25	25	25	30	30	30	30	30	30	30	30	30	30	30
45	45	45	45	45	45	45	40	40	40	40	40	40	40	40	40	40	40
45	45	45	45	45	45	45	30	30	30	30	30	30	30	30	30	30	30
80	20	80	20	80	80	80	80	80	80	80	80	80	80	80	80	80	80
20	80	20	80	20	20	20	20	20	20	20	20	20	20	20	20	20	20
20	20	20	20	20	20	20	20	80	80	80	80	80	80	80	80	80	80
80	80	80	80	80	80	80	80	20	20	20	20	20	20	20	20	20	20
80	80	80	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20
20	20	20	80	80	80	80	80	80	80	80	80	80	80	80	80	80	80
80	80	80	80	80	80	80	80	80	80	80	20	20	20	20	20	20	20
20	20	20	20	20	20	20	20	20	20	20	80	80	80	80	80	80	80
80	80	80	80	80	80	80	80	80	80	80	20	20	20	20	20	20	20
20	20	20	20	20	20	20	20	20	20	20	80	80	80	80	80	80	80
25	25	25	25	25	25	25	10	10	10	10	10	10	10	10	10	10	10
25	25	25	25	25	25	25	15	15	15	15	15	15	15	15	15	15	15
25	25	25	25	25	25	25	25	25	25	25	25	25	25	25	25	25	25
15	15	15	15	15	15	15	25	25	25	25	25	25	25	25	25	25	25
10	10	10	10	10	10	10	25	25	25	25	25	25	25	25	25	25	25
25	25	25	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10
25	25	25	15	15	15	15	15	15	15	15	15	15	15	15	15	15	15
25	25	25	25	25	25	25	25	25	25	25	25	25	25	25	25	25	25
15	15	15	25	25	25	25	25	25	25	25	25	25	25	25	25	25	25
10	10	10	25	25	25	25	25	25	25	25	25	25	25	25	25	25	25
50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50
50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50
50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50
50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50
100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50
50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50
50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50
50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50
50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50
50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50
50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50
50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50
50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50
50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50
50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50
50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50
50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50
50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50
50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50
50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50
50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50
50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50
30	70	30	70	30	70	30	70	30	70	30	70	30	70	30	70	30	70
70	30	70	30	70	30	70	30	70	30	70	30	70	30	70	30	70	30
50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50
50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50

Dodge_Ram_3500_Club_(4WD)	Ford_F150_(2WD)	Ford_F150_(4WD)	Ford_F150_Extended_(2WD)	Ford_F150_Extended_(4WD)	Ford_F250_(2WD)	Ford_F250_(4WD)	Ford_F250_Extended_(2WD)	Ford_F250_Extended_(4WD)	Ford_F350_(2WD)	Ford_F350_(4WD)	Ford_F350_Crew_Cab_(2WD)	Ford_F350_Crew_Cab_(4WD)	Ford_F350_Extended_(2WD)	Ford_Ranger_(2WD)	Ford_Ranger_(4WD)	Ford_Ranger_Extended_(2WD)	Ford_Ranger_Extended_(4WD)
20	20	20	20	20	20	20	20	20	20	20	20	20	20	60	60	60	60
60	60	60	60	60	60	60	60	60	60	60	60	60	60	20	20	20	20
20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20
1	10	1	1	1	10	1	1	1	1	1	1	1	1	60	1	10	1
1	20	9	9	1	20	1	1	1	9	1	1	1	1	20	9	20	9
1	40	20	20	1	40	1	8	1	20	1	1	1	1	15	20	40	20
1	20	40	40	7	20	7	20	1	40	7	7	1	7	1	40	20	40
1	7	20	20	20	7	20	40	16	20	20	20	1	20	1	20	7	20
15	1	8	8	40	1	40	20	20	8	40	40	15	40	1	8	1	8
20	1	1	1	20	1	20	9	40	1	20	20	20	20	1	1	1	1
60	1	1	1	10	1	10	1	20	1	10	10	60	10	1	1	1	1
35	65	35	65	35	65	80	65	35	65	35	65	35	65	65	35	65	35
65	35	65	35	65	35	65	35	65	35	65	35	65	35	35	65	35	65
30	50	30	50	30	50	30	50	30	50	30	50	30	50	30	30	70	30
70	50	70	50	70	50	70	50	70	50	70	50	70	50	30	70	30	70
35	35	35	35	35	35	35	35	35	35	35	35	35	35	50	50	50	50
65	65	65	65	65	65	65	65	65	65	65	65	65	65	50	50	50	50
35	35	35	35	35	35	35	35	35	35	35	35	35	35	50	50	50	50
65	65	65	65	65	65	65	65	65	65	65	65	65	65	50	50	50	50
35	35	35	35	35	35	35	35	35	35	35	35	35	35	50	50	50	50
65	65	65	65	65	65	65	65	65	65	65	65	65	65	50	50	50	50
10	25	25	10	10	25	25	10	10	25	25	10	10	10	25	25	10	10
15	25	25	15	15	25	25	15	15	25	25	15	15	15	25	25	15	15
25	25	25	25	25	25	25	25	25	25	25	25	25	25	25	25	25	25
25	15	15	25	25	15	15	25	25	15	15	25	25	25	15	15	25	25
25	10	10	25	25	10	10	25	25	10	10	25	25	25	10	10	25	25
40	40	40	40	40	40	40	40	40	40	40	40	40	40	40	40	20	40
40	40	40	40	40	40	40	40	40	40	40	40	40	40	40	40	40	40
20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	40	20
10	10	10	10	10	10	10	10	10	10	10	10	10	10	25	25	25	25
15	15	15	15	15	15	15	15	15	15	15	15	15	15	25	25	25	25
25	25	25	25	25	25	25	25	25	25	25	25	25	25	25	25	25	25
25	25	25	25	25	25	25	25	25	25	25	25	25	25	15	15	15	15
50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50
50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50
25	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50
75	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50
50	25	25	25	25	25	25	25	25	25	25	25	25	25	25	25	25	25
50	75	75	75	75	75	75	75	75	75	75	75	75	75	75	75	75	75
50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50
50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50
50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50
50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50
50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50
50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50
50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50
50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50
50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50
50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50
35	35	35	35	35	35	35	35	35	35	35	35	35	35	50	50	50	50
50	50	50	50	50	50	50	50	50	50	50	50	50	50	35	35	35	35
30	30	30	30	30	30	30	30	30	30	30	30	30	30	25	25	25	25
40	40	40	40	40	40	40	40	40	40	40	40	40	40	45	45	45	45
30	30	30	30	30	30	30	30	30	30	30	30	30	30	45	45	45	45
80	80	80	80	80	80	80	80	80	80	80	80	80	80	20	80	20	80
20	20	20	20	20	20	20	20	20	20	20	20	20	20	80	20	80	20
80	20	20	20	20	80	80	80	80	80	80	80	80	80	20	20	20	20
20	80	80	80	80	20	20	20	20	20	20	20	20	20	80	80	80	80
20	20	20	20	20	20	20	20	20	20	20	20	20	20	80	80	80	80
80	80	80	80	80	80	80	80	80	80	80	80	80	80	20	20	20	20
20	80	80	80	80	20	20	20	20	20	20	20	20	20	80	80	80	80
80	20	20	20	20	80	80	80	80	80	80	80	80	80	20	20	20	20
20	80	80	80	80	80	80	80	80	80	80	80	80	20	80	80	80	80
80	20	20	20	20	20	20	20	20	20	20	20	20	80	20	20	20	20
10	10	10	10	10	10	10	10	10	10	10	10	10	10	25	25	25	25
15	15	15	15	15	15	15	15	15	15	15	15	15	15	25	25	25	25
25	25	25	25	25	25	25	25	25	25	25	25	25	25	25	25	25	25
25	25	25	25	25	25	25	25	25	25	25	25	25	25	15	15	15	15
25	25	25	25	25	25	25	25	25	25	25	25	25	25	10	10	10	10
10	25	25	25	25	25	25	25	25	25	25	25	25	25	25	25	25	25
15	25	25	25	25	25	25	25	25	25	25	25	25	25	25	25	25	25
25	25	25	25	25	25	25	25	25	25	25	25	25	25	25	25	25	25
25	15	15	15	15	15	15	15	15	15	15	15	15	15	15	15	15	15
25	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10
50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50
50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50
50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50
50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50
100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50
50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50
50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50
50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50
50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50
50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50
50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50
50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50
50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50
50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50
50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50
50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50
50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50
50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50
50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50
50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50
50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50
50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50
30	70	30	70	30	70	30	70	30	70	30	70	30	70	70	30	70	30
70	30	70	30	70	30	70	30	70	30	70	30	70	30	30	70	30	70
50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50
50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50

GMC_Sierra_C1500_(2WD)	GMC_Sierra_C1500_Extended_(2WD)	GMC_Sierra_C2500_(2WD)	GMC_Sierra_C2500_Extended_(2WD)	GMC_Sierra_C3500_(2WD)	GMC_Sierra_C3500_Crew_Cab_(2WD)	GMC_Sierra_C3500_Extended_(2WD)	GMC_Sierra_K1500_(4WD)	GMC_Sierra_K1500_Extended_(4WD)	GMC_Sierra_K2500_(4WD)	GMC_Sierra_K2500_Extended_(4WD)	GMC_Sierra_K3500_(4WD)	GMC_Sierra_K3500_Crew_Cab_(4WD)	GMC_Sierra_K3500_Extended_(4WD)	GMC_Sonoma_(2WD)	GMC_Sonoma_(4WD)	GMC_Sonoma_Extended_(2WD)	GMC_Sonoma_Extended_(4WD)
20	20	20	20	20	20	20	20	20	20	20	20	20	20	60	60	60	60
60	60	60	60	60	60	60	60	60	60	60	60	60	60	20	20	20	20
20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20
10	1	1	1	1	1	1	1	1	1	1	1	1	1	60	1	10	1
20	1	9	1	1	1	1	1	1	1	1	1	1	1	20	9	20	1
40	8	20	8	8	1	1	8	1	1	1	1	1	1	15	20	40	8
20	20	40	20	20	7	10	20	7	7	7	7	1	1	1	40	20	20
7	40	20	40	40	20	20	40	20	20	20	20	1	1	1	20	7	40
1	20	8	20	20	40	40	20	40	40	40	40	40	15	15	1	8	1	20
1	9	1	9	9	20	20	9	9	20	20	20	20	20	1	1	1	9
1	1	1	1	1	10	10	1	1	10	10	10	60	60	1	1	1	1
65	65	65	65	65	65	65	35	35	35	35	35	35	35	65	35	65	35
35	35	35	35	35	35	35	65	65	65	65	65	65	65	35	65	35	65
50	50	50	50	50	50	50	30	30	30	30	30	30	30	70	30	70	30
50	50	50	50	50	50	50	70	70	70	70	70	70	70	30	70	30	70
35	35	35	35	35	35	35	35	35	35	35	35	35	35	50	50	50	50
65	65	65	65	65	65	65	65	65	65	65	65	65	65	50	50	50	50
35	35	35	35	35	35	35	35	35	35	35	35	35	35	50	50	50	50
65	65	65	65	65	65	65	65	65	65	65	65	65	65	50	50	50	50
35	35	35	35	35	35	35	35	35	35	35	35	35	35	50	50	50	50
65	65	65	65	65	65	65	65	65	65	65	65	65	65	50	50	50	50
25	10	25	10	25	10	10	25	10	25	10	25	10	10	25	25	10	10
25	15	25	15	25	15	15	25	15	25	15	25	15	15	25	25	15	15
25	25	25	25	25	25	25	25	25	25	25	25	25	25	25	25	25	25
15	25	15	25	15	25	25	15	25	15	25	15	25	25	15	15	25	25
10	25	10	25	10	25	25	10	25	10	25	10	25	25	10	10	25	25
40	40	40	40	40	40	40	40	40	40	40	40	40	40	40	40	20	40
40	40	40	40	40	40	40	40	40	40	40	40	40	40	40	40	40	40
20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	40	20
10	10	10	10	10	10	10	10	10	10	10	10	10	10	25	25	25	25
15	15	15	15	15	15	15	15	15	15	15	15	15	15	25	25	25	25
25	25	25	25	25	25	25	25	25	25	25	25	25	25	25	25	25	25
25	25	25	25	25	25	25	25	25	25	25	25	25	25	15	15	15	15
50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50
50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50
50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50
50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50
50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50
50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50
25	25	25	25	25	25	25	25	25	25	25	25	25	25	25	25	25	25
75	75	75	75	75	75	75	75	75	75	75	75	75	75	75	75	75	75
50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50
50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50
50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50
50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50
50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50
50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50
50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50
50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50
50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50
50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50
35	35	35	35	35	35	35	35	35	35	35	35	35	35	50	50	50	50
50	50	50	50	50	50	50	50	50	50	50	50	50	50	35	35	35	35
25	25	25	30	30	30	30	30	30	30	30	30	30	30	25	25	25	25
45	45	45	40	40	40	40	40	40	40	40	40	40	40	45	45	45	45
45	45	45	30	30	30	30	30	30	30	30	30	30	30	45	45	45	45
80	80	80	80	80	80	80	80	80	80	80	80	80	80	20	80	20	80
20	20	20	20	20	20	20	20	20	20	20	20	20	20	80	20	80	20
20	20	80	80	80	80	80	20	20	80	80	80	80	80	20	20	20	20
80	80	20	20	20	20	20	80	80	20	20	20	20	20	80	80	80	80
20	20	20	20	20	20	20	20	20	20	20	20	20	20	80	80	80	80
80	80	80	80	80	80	80	80	80	80	80	80	80	80	20	20	20	20
80	80	20	80	80	80	80	80	80	80	80	80	80	80	80	80	80	80
20	20	80	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20
80	80	80	80	80	80	80	80	20	20	20	20	20	20	80	80	80	80
20	20	20	20	20	20	20	20	80	80	80	80	80	80	20	20	20	20
10	10	10	10	10	10	10	10	10	10	10	10	10	10	25	25	25	25
15	15	15	15	15	15	15	15	15	15	15	15	15	15	25	25	25	25
25	25	25	25	25	25	25	25	25	25	25	25	25	25	25	25	25	25
25	25	25	25	25	25	25	25	25	25	25	25	25	25	15	15	15	15
25	25	25	25	25	25	25	25	25	25	25	25	25	25	10	10	10	10
25	25	25	25	25	25	25	25	25	25	25	25	25	25	25	25	25	25
25	25	25	25	25	25	25	25	25	25	25	25	25	25	25	25	25	25
25	25	25	25	25	25	25	25	25	25	25	25	25	25	25	25	25	25
15	15	15	15	15	15	15	15	15	15	15	15	15	15	15	15	15	15
10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10
50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50
50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50
50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50
50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50
100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50
50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50
50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50
50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50
50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50
50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50
50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50
50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50
50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50
50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50
50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50
50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50
50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50
70	70	70	70	70	70	70	30	30	30	30	30	30	30	70	30	70	30
30	30	30	30	30	30	30	70	70	70	70	70	70	70	30	70	30	70
50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50
50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50

Isuzu_Hombre	Isuzu_Hombre_Extended	Mazda_B2300_(2WD)	Mazda_B2300_Extended_(2WD)	Mazda_B4000_(4WD)	Mazda_B4000_Extended_(4WD)	Nissan_Truck_(2WD)	Nissan_Truck_(4WD)	Nissan_Truck_Extended_(2WD)	Nissan_Truck_Extended_(4WD)	Toyota_T100_(2WD)	Toyota_T100_Extended_(2WD)	Toyota_T100_Extended_(4WD)	Toyota_Tacoma_(2WD)	Toyota_Tacoma_(4WD)	Toyota_Tacoma_Extended_(2WD)	Toyota_Tacoma_Extended_(4WD)
60	60	60	60	60	60	60	60	60	60	20	20	20	60	60	60	60
20	20	20	20	20	20	20	20	20	20	60	60	60	20	20	20	20
20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20
60	10	60	10	1	1	60	1	10	1	10	1	1	20	1	10	1
20	20	20	20	9	1	20	9	20	1	20	1	1	40	9	20	1
15	40	15	40	20	8	15	20	40	8	40	8	1	20	20	40	8
1	20	1	20	40	20	1	40	20	20	20	20	1	16	40	20	20
1	7	1	7	20	40	1	20	7	40	7	40	16	1	20	7	40
1	1	1	1	8	20	1	8	1	20	1	20	20	1	8	1	20
1	1	1	1	1	9	1	1	1	9	1	9	40	1	1	1	9
1	1	1	1	1	1	1	1	1	1	1	1	20	1	1	1	1
65	65	65	65	35	35	65	35	65	35	65	65	35	65	35	65	35
35	35	35	35	65	65	35	65	35	65	35	35	65	35	65	35	65
30	70	70	70	30	90	70	30	70	30	50	50	30	70	30	70	30
30	30	30	30	70	70	30	70	30	70	50	50	70	30	70	30	70
50	50	50	50	50	50	50	50	50	50	35	35	35	50	50	50	50
50	50	50	50	50	50	50	50	50	50	65	65	65	50	50	50	50
50	50	50	50	50	50	50	50	50	50	35	35	35	50	50	50	50
50	50	50	50	50	50	50	50	50	50	65	65	65	50	50	50	50
50	50	50	50	50	50	50	50	50	50	35	35	35	50	50	50	50
50	50	50	50	50	50	50	50	50	50	65	65	65	50	50	50	50
10	10	25	10	25	10	25	25	10	10	25	10	10	25	25	10	10
15	15	25	15	25	15	25	25	15	15	25	15	15	25	25	15	15
25	25	25	25	25	25	25	25	25	25	25	25	25	25	25	25	25
15	25	15	25	15	25	15	15	25	25	15	25	25	15	15	25	25
10	25	10	25	10	25	10	10	25	25	10	25	25	10	10	25	25
40	20	40	20	40	40	40	40	20	40	40	40	40	40	40	20	40
40	40	40	40	40	40	40	40	40	40	40	40	40	40	40	40	40
20	40	20	40	20	20	20	20	40	20	20	20	20	20	20	40	20
25	25	25	25	25	25	25	25	25	25	10	10	10	25	25	25	25
25	25	25	25	25	25	25	25	25	25	15	15	15	25	25	25	25
25	25	25	25	25	25	25	25	25	25	25	25	25	25	25	25	25
15	15	15	15	15	15	15	15	15	15	25	25	25	15	15	15	15
50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50
50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50
50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50
50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50
50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50
50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50
50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50
50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50
25	25	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50
75	75	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50
50	50	25	25	25	25	50	50	50	50	50	50	50	50	50	50	50
50	50	75	75	75	75	50	50	50	50	50	50	50	50	50	50	50
50	50	50	50	50	50	25	25	25	25	50	50	50	50	50	50	50
50	50	50	50	50	50	75	75	75	75	50	50	50	50	50	50	50
50	50	50	50	50	50	50	50	50	50	25	25	25	25	25	25	25
50	50	50	50	50	50	50	50	50	50	75	75	75	75	75	75	75
50	50	50	50	50	50	50	50	50	50	35	35	35	50	50	50	50
35	35	35	35	35	35	35	35	35	35	50	50	50	35	35	35	35
25	25	25	25	25	25	25	25	25	25	30	30	30	25	25	25	25
45	45	45	45	45	45	45	45	45	45	40	40	40	45	45	45	45
45	45	45	45	45	45	45	45	45	45	30	30	30	45	45	45	45
20	20	20	20	80	80	20	20	20	20	20	80	80	20	20	20	20
80	80	80	80	20	20	80	80	80	80	80	20	20	80	80	80	80
80	20	80	80	20	20	80	80	80	80	80	20	20	80	20	20	20
20	80	20	20	80	80	20	20	20	20	20	80	80	20	80	80	80
80	80	80	80	80	80	80	80	80	80	80	80	80	80	80	80	80
20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20
80	80	80	80	80	80	80	80	80	80	80	80	80	80	80	80	80
20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20
80	80	80	80	80	80	80	80	80	80	80	80	80	80	80	80	80
20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20
25	25	25	25	25	25	25	25	25	25	10	10	10	25	25	25	25
25	25	25	25	25	25	25	25	25	25	15	15	15	25	25	25	25
25	25	25	25	25	25	25	25	25	25	25	25	25	25	25	25	25
15	15	15	15	15	15	15	15	15	15	25	25	25	15	15	15	15
10	10	10	10	10	10	10	10	10	10	25	25	25	10	10	10	10
25	25	25	25	25	25	25	25	25	25	25	25	25	25	25	25	25
25	25	25	25	25	25	25	25	25	25	25	25	25	25	25	25	25
25	25	25	25	25	25	25	25	25	25	25	25	25	25	25	25	25
15	15	15	15	15	15	15	15	15	15	15	15	15	15	15	15	15
10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10
50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50
50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50
50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50
50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50
100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50
50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50
50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50
50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50
50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50
50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50
50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50
50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50
50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50
50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50
50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50
50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50
50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50
50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50
50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50
50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50
50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50
50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50
70	70	70	70	30	30	70	30	70	30	70	70	30	70	30	70	30
30	30	30	30	70	70	30	70	30	70	30	30	70	30	70	30	70
50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50
50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50

G. Sales Features and Problems Database

key	Sales Feaure	Feature Weight	Sales Problem	problem Weight
compact	This is a compact truck and you stated that you prefer compact trucks	.5	This is a full size truck and you stated that you prefer compact trucks	.5
full	This is a full size truck and you stated that you prefer full size trucks	.5	This is a compact truck and you stated that you prefer full size trucks	.5
either	Either Truck	0	No_Either Truck	0.0
Price_10_12K	This truck is within your preferred price range	.5	This truck is not within your preferred price range	.5
Price_12_14K	This truck is within your preferred price range	.5	This truck is not within your preferred price range	.5
Price_14_16K	This truck is within your preferred price range	0.5	This truck is not within your preferred price range	0.5
Price_16_18K	This truck is within your preferred price range	.5	This truck is not within your preferred price range	.5
Price_18_20K	This truck is within your preferred price range	.5	This truck is not within your preferred price range	.5
Price_20_22K	This truck is within your preferred price range	.5	This truck is not within your preferred price range	.5
Price_22_24K	This truck is within your preferred price range	.5	This truck is not within your preferred price range	.5
Price_Over_24K	This truck is within your preferred price range	0.5	This truck is not within your preferred price range	0.5
Off_road_no	You said that you prefer trucks that are not designed for offroad driving	0	No_You said that you prefer trucks that are not designed for offroad driving	0
Off_road_yes	You said that you prefer trucks that are designed for offroad driving	.5	This truck is not designed for offroad driving	.5
Towing_no	You said that you prefer trucks that are not designed for towing	0	NO_You said that you prefer trucks that are not designed for towing	0
Towing_yes	You said that you prefer trucks that are designed for towing	.5	This truck is not designed for towing	.5
Hauling_no	You said that you prefer trucks that are not designed for hauling things	0	NO_You said that you prefer trucks that are not designed for hauling things	0
Hauling_yes	You said that you prefer trucks that are designed for hauling things.	.5	This truck is not designed for hauling things.	.5
Construction_no	You said that you prefer trucks that are not designed for construction work	0	You said that you prefer trucks that are not designed for construction work	0
Construction_yes	You said that you prefer trucks that are designed for construction work	.5	This truck is not designed for construction work	.5
Plowing_Snow_no	You said that you prefer trucks that are not designed for plowing snow	0	You said that you prefer trucks that are not designed for plowing snow	0
Plowing_Snow_yes	You said that you prefer trucks that are designed for plowing snow	0.5	This truck is not designed for plowing snow	0.5
Passengers_1	You said that you prefer trucks that are good for carrying a small number of passengers	.25	You said that you prefer trucks that are good for carrying a small number of passengers	0
Passengers_2	You said that you prefer trucks that are good for carrying a small number of passengers	.25	You said that you prefer trucks that are good for carrying a small number of passengers	0
Passengers_3	You said that you prefer trucks that are good for carrying a small number of passengers	.25	You said that you prefer trucks that are good for carrying a small number of passengers	0
Passengers_4	You said that you prefer trucks that are good for carrying a large number of passengers	0.5	You said that you prefer trucks that are good for carrying a large number of passengers	0
Passengers_5	You said that you prefer trucks that are good for carrying a large number of passengers	.5	This truck isn't very good for carrying a large number of passengers	.5
Passengers_front_seat_1	You prefer trucks that are good for carrying a small number of passengers in the front seat	.25	You prefer trucks that are good for carrying a small number of passengers in the front seat	0
Passengers_front_seat_2	You prefer trucks that are good for carrying a small number of passengers in the front seat	.25	You prefer trucks that are good for carrying a small number of passengers in the front seat	0
Passengers_front_seat_3	You prefer trucks that are good for carrying a larger number of passengers in the front seat	.5	This truck isn't very good for carrying a large number of passengers in the front seat	.5
Biggest_available_1	You said that you would rather have the smallest available truck	.5	You said that you would rather have the smallest available truck	0
Biggest_available_2	You said that you would rather have the smallest available truck	.25	You said that you would rather have the smallest available truck	0
Biggest_available_3	You said that you didn't care if a truck was the smallest or biggest available truck	0	You said that you didn't care if a truck was the smallest or biggest available truck	0
Biggest_available_4	You said that you would rather have the biggest available truck	.25	You said that you would rather have the biggest available truck	0
Biggest_available_5	You said that you would rather have the biggest available truck	.5	You said that you would rather have the biggest available truck	0
Chevy_no	You said that you don't like Chevrolet trucks	0	You said that you don't like Chevrolet trucks	.5
Chevy_yes	You said that you like Chevrolet trucks	.5	You said that you like Chevrolet trucks	0
Dodge_no	You said that you don't like Dodge trucks	0	You said that you don't like Dodge trucks	.5
Dodge_yes	You said that you like Dodge trucks	.5	You said that you like Dodge trucks	0
Ford_no	You said that you don't like Ford trucks	0	You said that you don't like Ford trucks	.5
Ford_yes	You said that you like Ford trucks	.5	You said that you like Ford trucks	0
GMC_no	You said that you don't like GMC trucks	0	You said that you don't like GMC trucks	.5
GMC_yes	You said that you like GMC trucks	.5	You said that you like GMC trucks	0
Isuzu_no	You said that you don't like Isuzu trucks	0	You said that you don't like Isuzu trucks	.5
Isuzu_yes	You said that you like Isuzu trucks	.5	You said that you like Isuzu trucks	0
Mazda_no	You said that you don't like Mazda trucks	0	You said that you don't like Mazda trucks	.5
Mazda_yes	You said that you like Mazda trucks	.5	You said that you like Mazda trucks	0
Nissan_no	You said that you don't like Nissan trucks	0	You said that you don't like Nissan trucks	.5
Nissan_yes	You said that you like Nissan trucks	.5	You said that you like Nissan trucks	0
Toyota_no	You said that you don't like Toyota trucks	0	You said that you don't like Toyota trucks	.5
Toyota_yes	You said that you like Toyota trucks	.5	You said that you like Toyota trucks	0
Bed_length_short	You said that you like short bed trucks	.5	You said that you like short bed trucks	0
Bed_length-long	You said that you like long bed trucks	.5	You said that you like long bed trucks	0
height_Under_6	This truck is good for people under six feet tall	.5	This truck is not very good for people under six feet tall	.25
height_between_6_65	This truck is good for people over six feet tall	0.5	This truck is not very good for people over six feet tall	0
height_over_65	This truck is good for people over six feet tall	.5	This truck is not very good for people over six feet tall	.25
engine_size_4_no	This truck is not available with a 4 cylinder engine	0	This truck is available with a 4 cylinder engine	0
engine_size_4_yes	This truck is available with a 4 cylinder engine	.5	This truck is not available with a 4 cylinder engine	.5
engine_size_6_no	This truck is not available with a 6 cylinder engine	0	This truck is available with a 6 cylinder engine	0
engine_size_6_yes	This truck is available with a 6 cylinder engine	.5	This truck is not available with a 6 cylinder engine	.5
engine_size_8_no	This truck is not available with an 8 cylinder engine	0	This truck is available with an 8 cylinder engine	0
engine_size_8_yes	This truck is available with an 8 cylinder engine	.5	This truck is not available with an 8 cylinder engine	.5
engine_size_10_no	This truck is not available with a 10 cylinder engine	0	This truck is available with a 10 cylinder engine	0
engine_size_10_yes	This truck is available with a 10 cylinder engine	.5	This truck is not available with a 10 cylinder engine	.5
engine_size_8_diesel_no	This truck is not available with an 8 cylinder diesel engine	0	This truck is available with an 8 cylinder diesel engine	0
engine_size_8_diesel_yes	This truck is available with an 8 cylinder diesel engine	.5	This truck is not available with an 8 cylinder diesel engine	0.5
Big_quiet_comfortable_1	You said that you don't prefer a big, quiet, comfortable truck	0	You said that you don't prefer a big, quiet, comfortable truck	0
Big_quiet_comfortable_2	You said that you don't prefer a big, quiet, comfortable truck	0	You said that you don't prefer a big, quiet, comfortable truck	0
Big_quiet_comfortable_3	You said that you prefer a big, quiet, comfortable truck	0	You said that you prefer a big, quiet, comfortable truck	0
Big_quiet_comfortable_4	You said that you prefer a big, quiet, comfortable truck	0	You said that you prefer a big, quiet, comfortable truck	0
Big_quiet_comfortable_5	You said that you don't prefer a big, quiet, comfortable truck	0	You said that you don't prefer a big, quiet, comfortable truck	0
traditional_futuristic_1	You said that you prefer a traditional-looking truck	.5	This isn't a traditional-looking truck	.5
traditional_futuristic_2	You said that you prefer a traditional-looking truck	0	You said that you prefer a traditional-looking truck	0
traditional_futuristic_3	You said that you prefer a traditional-looking truck	0	You said that you prefer a traditional-looking truck	0
traditional_futuristic_4	You said that you prefer a futuristic-looking truck	0	You said that you prefer a futuristic-looking truck	0
traditional_futuristic_5	You said that you prefer a futuristic-looking truck	0.5	This isn't a futuristic-looking truck	.5
2wd	You said that you prefer a two-wheel drive truck	.5	This isn't a two-wheel drive truck	.25
4wd	You said that you prefer a four-wheel drive truck	.5	This isn't a four-wheel drive truck	.25